

<b>Document Title</b>	Software Component Template
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	62

<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R21-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Description</b>
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Introduction of imposition times for constraints</li> <li>• Clean-up of diagnostics service needs</li> <li>• New writing strategy for NvRAM</li> <li>• minor corrections / clarifications / editorial changes</li> </ul>
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• In-place Strategy for Array Data Type</li> <li>• New Service Use Cases</li> <li>• Optimization of Return Type for RTE APIs</li> <li>• minor corrections / clarifications / editorial changes</li> </ul>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Support for optimized access to coding data</li> <li>• Support for meta-data on application level</li> <li>• Support for optimized storing of bitfields in NvRAM</li> <li>• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> <li>• Changed Document Status from Final to published</li> </ul>

2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Support for optional elements in structured data types</li> <li>• Improved description of service use cases</li> <li>• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Improved support for Unions</li> <li>• Improved upstream mapping</li> <li>• Improved description of service use cases</li> <li>• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Efficient NV data handling</li> <li>• Introduction of data transformation</li> <li>• Support for variable-size Arrays of arbitrary data types</li> <li>• Support for ASIL/QM development</li> <li>• Minor corrections / clarifications / editorial changes; For details please refer to the BWCStatement</li> </ul>
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Various fixes and clarifications</li> </ul>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Various fixes and clarifications</li> </ul>

2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Introduction of <a href="#">PRPortPrototype</a></li> <li>• Definition of implicit communication behavior</li> <li>• Support for the formal analysis of resource locking</li> <li>• Introduction of refined scheduling of <a href="#">RunnableEntityS</a></li> <li>• Get information about activating <a href="#">RTEEvent</a></li> <li>• Connection of Mode Managers and Mode Users with different number of <a href="#">ModeDeclarations</a></li> <li>• Support activation of <a href="#">RunnableEntityS</a> on remote ECUs</li> <li>• Support for <a href="#">ModeTransition</a></li> <li>• Support for the definition of the network representation of composite data types</li> <li>• <a href="#">ServiceNeeds</a> for diagnostics over IP</li> <li>• Various fixes and clarifications</li> </ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Added <code>CompuMethod</code> categories <code>SCALE_LINEAR_AND_TEXTTABLE</code> and <code>SCALE_RATIONAL_AND_TEXTTABLE</code> (table <a href="#">5.80</a>)</li> <li>• Clarification concerning the usage of invalid values</li> <li>• Revised support for data filters</li> <li>• Support for partial networking</li> <li>• Support for the specification of local connections between software-components</li> <li>• Improved description of service needs</li> <li>• Change history of constraints and specification items</li> <li>• Miscellaneous improvements and clarifications</li> <li>• “Support for Standardization” moved to Standardization Template <a href="#">[1]</a></li> </ul>

2011-04-15	4.0.2	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Remove restriction on data type of inter-runnable variables</li> <li>• Rework end-to-end communication protection</li> <li>• Add more constraints on the usage of the meta-model</li> <li>• Various fixes and clarifications</li> </ul>
2009-12-18	4.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• New requirements tracing table</li> <li>• Support for fixed data exchange</li> <li>• Implementation of meta-model cleanup</li> <li>• Fundamental revision of the data type concept</li> <li>• Support for variant handling</li> <li>• Support for end-to-end communication protection</li> <li>• Support for documentation</li> <li>• Support for stopping and restarting of software-components</li> <li>• Support for triggered events</li> <li>• Support for explicit mapping of interface elements</li> <li>• Revised concept of mode management</li> <li>• Support for integrity and scaling at ports</li> <li>• Support for standardization within AUTOSAR</li> </ul>
2008-08-13	3.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Improved support for on-board diagnostics</li> <li>• Small layout adaptations made</li> </ul>
2007-12-21	3.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Improved support for measurement and calibration</li> <li>• Improved semantics of delegation ports</li> <li>• Introduction of abstract memory classes</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>

2007-01-24	2.1.15	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Harmonization of the document with other specifications (e.g. RTE)</li> <li>• Introduction of a new concept to support calibration and measurement - harmonized with RTE</li> <li>• 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)</li> <li>• Legal disclaimer revised</li> <li>• Release notes added</li> <li>• “Advice for users” added</li> <li>• “Revision information” added</li> </ul>
2006-05-16	2.0.0	AUTOSAR Administration	Second
2005-05-31	1.0.0	AUTOSAR Administration	Initial release



## Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

1	Introduction	29
1.1	Overview	29
1.2	Scope	29
1.3	Organization of the Meta-Model	30
1.4	Structure of the Template	32
1.4.1	Description of Software Components on VFB Level	32
1.4.2	Description of Software Components on RTE Level	32
1.4.3	Descriptions of Software Components on Implementation Level	33
1.5	Abbreviations	33
1.6	Document Conventions	34
1.7	Imposition Times of Constraints	36
1.8	Requirements Tracing	38
2	Conceptual Aspects	46
2.1	Introduction	46
2.2	Measurement and Calibration	46
2.2.1	Basic Approach of Measurement and Calibration	46
2.2.2	Calibration Parameters Overview	46
2.2.3	Using Calibration Parameters	47
2.2.3.1	Sharing Calibration Parameters within Compositions	47
2.2.3.2	Sharing Calibration Parameters between SwComponentPrototypes of the Same SwComponentType	49
2.2.3.3	Providing Instance Individual Characteristic Data	50
2.3	Runtime and Data Consistency Aspects	51
2.3.1	Background: the Issues	51
2.3.1.1	Mutual Exclusion with Semaphores	51
2.3.1.2	Interrupt Disabling	52
2.3.1.3	Priority Ceiling	52
2.3.1.4	Implicit Communication by Means of Variable Copies	52
2.3.2	Data Consistency at Runtime	53
2.3.3	Modeling Aspects of Data Consistency	54
2.4	Variant Handling in the Software Component Template	55
2.5	Communication Specification of Composition Component Types	57
2.5.1	Rationale	57
2.6	PRPortPrototype	58
2.6.1	Use Case 1	58
2.6.2	Use Case 2	59
2.6.3	Use Case 3	60
2.6.4	Solution	60
2.7	Variable-size Array Data Types	61
2.7.1	Overview and Use cases	61
2.7.1.1	“Old-world” dynamic-size Arrays	61
2.7.1.2	“New-world” variable-size Arrays	63
2.7.2	Modeling Aspects regarding Application Data Types	65



2.7.3	Modeling Aspects regarding Implementation Data Types . . .	66
2.8	Optional Elements in Structures . . . . .	67
2.8.1	Background . . . . .	67
3	Overview: Software Components, Ports, and Interfaces . . . . .	69
3.1	Introduction . . . . .	69
3.2	Software Component . . . . .	70
3.2.1	Overview . . . . .	70
3.2.2	PortPrototype . . . . .	71
3.2.3	AtomicSwComponentType . . . . .	75
3.2.4	ParameterSwComponentType . . . . .	77
3.2.5	Symbolic Name of a Software-Component . . . . .	78
3.3	Composition . . . . .	79
3.3.1	Overview . . . . .	79
3.3.2	SwComponentPrototype . . . . .	80
3.3.3	Connectors . . . . .	83
3.3.4	Instantiation-specific RTEEvents . . . . .	88
3.4	Port Interface . . . . .	90
4	Details: Software Components, Ports, and Interfaces . . . . .	96
4.1	Introduction . . . . .	96
4.2	Port Interface Details . . . . .	96
4.2.1	Introduction . . . . .	96
4.2.2	Sender Receiver Communication . . . . .	97
4.2.2.1	Sender Receiver Interface . . . . .	97
4.2.2.2	Meta-data on the Application Software Level . . . . .	101
4.2.3	Client Server Communication . . . . .	103
4.2.3.1	Client Server Interface . . . . .	103
4.2.3.2	Error Handling in Client/Server Communication . . . . .	109
4.2.4	External Trigger Event Communication . . . . .	111
4.2.5	Communication of Modes . . . . .	114
4.2.6	Parameter Communication . . . . .	119
4.3	PortInterface Mapping and Data Scaling . . . . .	120
4.3.1	PortInterface Mapping . . . . .	122
4.3.1.1	Mapping of Sender Receiver Interface, Parameter In- terface and Non Volatile Data Interface Elements . . . . .	125
4.3.1.2	Mapping of Client Server Interface Elements . . . . .	128
4.3.1.3	Mapping of Mode Interface Elements . . . . .	131
4.3.1.4	Mapping of Trigger Interface Elements . . . . .	135
4.3.1.5	Mapping of Elements of a composite Data Type . . . . .	136
4.3.2	Data Conversion . . . . .	142
4.3.2.1	Linear Data Scaling . . . . .	143
4.3.2.2	Table Conversion . . . . .	144
4.3.3	Relevance for Data Transformation . . . . .	149
4.4	Port Annotation . . . . .	152
4.4.1	Introduction . . . . .	152
4.4.2	SenderReceiverAnnotation . . . . .	154

4.4.3	ClientServerAnnotation	157
4.4.4	Annotation for the I/O Hardware Abstraction Layer	158
4.4.5	Parameter Port Annotation	160
4.4.6	Mode Port Annotation	161
4.4.7	Trigger Port Annotation	162
4.4.8	Non Volatile Data Port Annotation	162
4.4.9	Delegated Port Annotations	163
4.4.10	General Annotation	164
4.5	Communication Specification	165
4.5.1	Communication Specification for Sender-Receiver Communication	168
4.5.1.1	Receiver ComSpec	169
4.5.1.2	Sender ComSpec	176
4.5.1.3	Data Filter	181
4.5.1.4	Communication between Application and NV Block Software Components	183
4.5.1.5	Communication behavior to be implemented by the Software Component	185
4.5.2	Communication Specification for Client-Server Communication	186
4.5.2.1	Client ComSpec	186
4.5.2.2	Server ComSpec	187
4.5.3	Communication Specification for Mode Switch Communication	189
4.5.3.1	Mode Switch Sender ComSpec	189
4.5.3.2	Mode Switch Receiver ComSpec	190
4.5.4	Communication Specification for Parameters	191
4.5.5	Communication Specification for NV Data	193
4.5.5.1	NV Require ComSpec	193
4.5.5.2	NV Provide ComSpec	194
4.5.6	Configuration of Data Transformation	195
4.6	Port Groups within Component Types	201
4.7	End to End Protection	202
4.8	Partial Networking	214
4.8.1	VFC Control Ports	215
4.8.2	VFC Status Ports	215
4.9	Formal Definition of implicit Communication Behavior	216
4.9.1	Consistency Needs on Receiver Side	221
4.9.2	Consistency Needs on Sender Side	222
4.9.3	Consistency Needs for Senders and receivers of the same Data inside on RunnableEntityGroup	222
5	Data Description	223
5.1	Introduction	223
5.2	Data Types	227
5.2.1	Overview	227
5.2.2	Data Type Mapping	229
5.2.3	Data Categories	233

5.2.4	Application Data Type . . . . .	237
5.2.4.1	Application Primitive Data Types . . . . .	239
5.2.4.2	Application Composite Data Types . . . . .	253
5.2.5	Implementation Data Type . . . . .	267
5.2.5.1	Overview . . . . .	267
5.2.5.2	Modeling of Type Reference using Implementation Data Type . . . . .	273
5.2.5.3	Modeling of Structure using Implementation Data Type . . . . .	274
5.2.5.4	Modeling of Union using Implementation Data Type . . . . .	276
5.2.5.5	Modeling of Array using Implementation Data Type . . . . .	277
5.2.5.6	Modeling of Pointer using Implementation Data Type . . . . .	288
5.2.6	Base Type . . . . .	292
5.2.7	Data Type Terminology . . . . .	299
5.2.7.1	Primitive Type . . . . .	299
5.2.7.2	Compound Primitive Data Type . . . . .	299
5.2.7.3	Integral Primitive Type . . . . .	300
5.2.7.4	Variable-Size Array Data Type . . . . .	301
5.2.7.5	Wrapped Union Data Type . . . . .	301
5.2.7.6	Optional Element Structure . . . . .	304
5.3	Data Prototypes . . . . .	305
5.3.1	Overview . . . . .	305
5.3.2	Data Constraints for DataPrototypes typed by Array DataTypes . . . . .	311
5.3.3	Reference to Data Prototypes . . . . .	315
5.3.3.1	AUTOSAR Variable Ref . . . . .	315
5.3.3.2	AUTOSAR Parameter Ref . . . . .	316
5.3.3.3	Modeling Approach . . . . .	318
5.3.3.4	Access into VariableDataPrototype typed by an ImplementationData Type . . . . .	321
5.3.3.5	Access into ParameterDataPrototype typed by an ImplementationData Type . . . . .	324
5.4	Properties of Data Definitions . . . . .	326
5.4.1	Overview . . . . .	326
5.4.2	Invalid Value . . . . .	340
5.4.3	Properties for Measurement . . . . .	345
5.4.4	Properties of Curves and Maps . . . . .	347
5.4.4.1	Specification of grouped Axes . . . . .	355
5.4.4.2	Specification of fix Axes . . . . .	358
5.4.5	Setting an Axis Input Value . . . . .	361
5.4.6	Setting a Group Axis . . . . .	366
5.4.7	Specifying Data Dependencies . . . . .	371
5.4.8	Precedence of data properties with respect to data elements, axis elements, computation methods, units . . . . .	373
5.5	Elements used in Properties of Data Definitions . . . . .	378
5.5.1	Computation Methods . . . . .	379
5.5.1.1	Category Values in the context of a CompuMethod . . . . .	389

5.5.1.2	Applicability of Attributes in the context of a CompuMethod . . . . .	390
5.5.1.3	CompuMethod and AutosarDataType . . . . .	392
5.5.1.4	Example for Enumeration . . . . .	394
5.5.1.5	Example for Linear Conversion . . . . .	395
5.5.1.6	Example for Linear Conversion with texttable . . . . .	395
5.5.1.7	Example for conversion specified by a rational function . . . . .	396
5.5.1.8	Example for BITFIELD_TEXTTABLE . . . . .	397
5.5.2	Physical Units, Physical Dimensions and Unit Groups . . . . .	400
5.5.3	Data Constraints . . . . .	407
5.5.3.1	Physical Limits . . . . .	414
5.5.4	Addressing Methods . . . . .	415
5.5.5	Record Layouts . . . . .	423
5.5.5.1	Specifying Record Layouts . . . . .	424
5.5.5.2	RecordLayouts and DataTypes . . . . .	432
5.5.5.3	Record Layouts and Interpolation Routines . . . . .	439
5.5.6	Display Presentation . . . . .	440
5.6	Specification of Constant Values . . . . .	442
5.6.1	Overview . . . . .	442
5.6.2	Reference to Constant . . . . .	448
5.6.3	Values for Variable-Size Array . . . . .	449
5.6.4	Values for Compound Primitive Data Types . . . . .	451
5.6.5	Values for BITFIELD_TEXTTABLE . . . . .	459
5.6.6	Specification of Values based on Rules . . . . .	461
5.6.6.1	Support for primitive Data Types . . . . .	461
5.6.6.2	Support for composite Data Types . . . . .	470
5.6.6.3	Support for compound primitive Data Types . . . . .	479
5.6.7	Examples . . . . .	480
5.6.7.1	Example for Constant Specification for CURVE . . . . .	480
5.6.7.2	Example for Constant Specification for MAP . . . . .	481
5.6.7.3	Example for Constant Specification for MAP with two STD_AXIS . . . . .	483
5.6.7.4	Example for Constant Specification for COM_AXIS . . . . .	484
5.6.7.5	Example for Constant Specification for an Array of compound primitive Objects . . . . .	485
5.7	Initial Values . . . . .	487
5.7.1	Overview . . . . .	487
5.7.2	Initial Value Representation . . . . .	488
5.7.3	Constant Specification Mapping . . . . .	490
5.7.4	Initial Values For CalibrationParameters . . . . .	492
5.7.5	Initial Value for optional Element . . . . .	494
5.7.5.1	Initial Value for optional ApplicationRecordElement . . . . .	494
5.7.5.2	Initial Value for optional ImplementationDataType-Element . . . . .	495
6	Compatibility . . . . .	496

6.1	Introduction	496
6.2	Compatibility of Data Types	496
6.2.1	ApplicationDataType	496
6.2.1.1	ApplicationPrimitiveDataType	496
6.2.1.2	ApplicationCompositeDataType	497
6.2.2	ImplementationDataType	498
6.2.3	Compatibility of SwBaseType	500
6.2.4	Compatibility of SwDataDefProps	500
6.2.4.1	Compatibility of Units	501
6.2.4.2	Compatibility of PhysicalDimensions	502
6.2.4.3	Compatibility of Data Constraints	503
6.2.4.4	Compatibility in case of ImplementationDataType	504
6.2.4.5	Compatibility of CompuMethods	505
6.2.4.6	Compatibility of Record Layouts	507
6.2.5	Compatibility of ApplicationDataType and Implementation- DataType	508
6.3	Compatibility of Variable Data Prototypes and Parameter Data Prototypes	512
6.4	Compatibility of Sender Receiver Interfaces, Parameter Interfaces and Non Volatile Data Interfaces	514
6.4.1	Connection of Required and Provided Port via Assem- blySwConnector	514
6.4.2	Connection of Inner and Outer Port via DelegationSwCon- nector	515
6.4.3	Connection of Required and Provided Port via PassThrough- SwConnector	516
6.4.4	Compatibility of ParameterDataPrototype and VariableDat- aPrototype depending on PortInterface Type	517
6.5	Compatibility of Mode Switch Interfaces	518
6.5.1	Connection of Required and Provided Port via Assem- blySwConnector	518
6.5.2	Connection of Inner and Outer Port via DelegationSwCon- nector	519
6.5.3	Connection of Outer and Outer Port via PassThroughSwCon- nector	519
6.6	Compatibility of Mode Declaration Group Prototypes	520
6.7	Compatibility of Mode Declaration Groups	520
6.8	Compatibility of Argument Prototypes	521
6.9	Compatibility of Application Errors	522
6.10	Compatibility of Client/Server Operations	522
6.11	Compatibility of Client Server Interfaces	523
6.11.1	Connection of Required and Provided Port via Assem- blySwConnector	523
6.11.2	Connection of Inner and Outer Port via DelegationSwCon- nector	523
6.11.3	Connection of Outer and Outer Port via PassThroughSwCon- nector	524

6.12	Compatibility of Trigger Interfaces	525
6.12.1	Connection of Required and Provided Port via AssemblySwConnector	525
6.12.2	Connection of Inner and Outer Port via DelegationSwConnector	525
6.12.3	Connection of Outer and Outer Port via PassThroughSwConnector	526
6.13	Compatibility of Trigger	526
6.14	Entire Delegation of a Provided Port Prototype	527
6.14.1	Split and Merge of PortInterface Elements	528
6.15	Compatibility in Case of a Flat ECU Extract	528
6.16	Compatibility Examples	529
6.16.1	Compatibility on Assembly Level	530
6.16.1.1	Legal Use	530
6.16.1.2	Illegal Use	531
6.16.2	Compatibility on Delegation Level	531
6.16.2.1	Legal Use	531
6.16.2.2	Illegal Use	534
7	Internal Behavior	536
7.1	Introduction	536
7.2	Runnable Entity	542
7.2.1	Concurrency and Reentrancy of a RunnableEntity that cannot be Invoked Concurrently	549
7.2.2	Concurrency and Reentrancy of a RunnableEntity that can be Invoked Concurrently	550
7.2.3	Timed Activation of Runnable Entities	551
7.2.4	Additional Remarks and Clarifications	552
7.2.4.1	Reentrancy and Multiple Instantiation	552
7.2.4.2	Reentrancy and “Library Functions”	553
7.2.4.3	Compatibility of ClientServerOperations triggering the same RunnableEntity	553
7.2.4.4	Categories of Runnable Entities	554
7.2.4.5	Arguments of a Runnable Entity	555
7.2.5	Activation Reason of a Runnable Entity	557
7.2.6	Runnable Entity for Initialization Purpose	559
7.3	RTEEvent	561
7.3.1	Defining an Event	567
7.3.2	Defining how to Respond to an Event	570
7.4	Communication among Runnable Entities	572
7.4.1	Description Possibility 1: Exclusive Area	573
7.4.1.1	Entire Runnable Runs in the Exclusive Area	576
7.4.1.2	Runnable would Dynamically Enter and Leave the Exclusive Area	576
7.4.1.3	Configuration of API Generation	577
7.4.2	Description Possibility 2: Inter-Runnable Variable	578

7.4.3	Inter Runnable Triggering	581
7.5	Data Access of RunnableEntities	582
7.5.1	RunnableEntities and Sender Receiver Communication	586
7.5.1.1	Terminology	586
7.5.1.2	Data Access	587
7.5.1.3	Explicit Sending and Receiving	589
7.5.1.4	Implicit Sending and Receiving	594
7.5.1.5	DataSendCompletedEvent	596
7.5.1.6	DataWriteCompletedEvent	596
7.5.1.7	DataReceivedEvent	597
7.5.1.8	DataReceiveErrorEvent	598
7.5.2	RunnableEntities and Client Server Communication	600
7.5.2.1	Invoking an Operation	600
7.5.2.2	Providing an Implementation of an Operation	605
7.5.2.3	Reacting on Data Transformation Errors	606
7.5.3	RunnableEntities and External Trigger Event Communication	606
7.5.3.1	Trigger Source	606
7.5.3.2	Trigger Sink	608
7.5.4	RunnableEntities and Parameter Access	608
7.5.4.1	InstantiationDataDefProps	610
7.5.5	RunnableEntities and Mode Communication	612
7.6	Port API Options	612
7.6.1	Enable to Take Address	614
7.6.2	Indirect API Generation	615
7.6.3	Port Defined Argument Value	615
7.6.4	Supported Features	616
7.6.4.1	Buffer Locking	617
7.7	PerInstanceMemory	618
7.7.1	PerInstanceMemory typed by "C" Data Types	619
7.7.2	PerInstanceMemory typed by AUTOSAR Data Types	620
7.8	Static Memory and Constant Memory	621
7.9	Included AUTOSAR Data Types	622
7.10	Included Mode Declaration Groups	623
7.11	Service Needs	624
7.11.1	Overview	624
7.11.2	Assignment of Service Needs to Ports and Data	626
7.12	Variation Point Proxy	635
8	Implementation	641
9	Mode Management	646
9.1	Declaration of Modes	646
9.2	Modes and Events	651
9.3	Initialization / Finalization	655
9.4	Mode Error Behavior	656
9.5	Summary Meta-Model Excerpt Related to Modes	659

10 ECU Abstraction and Complex Drivers	661
10.1 Introduction	661
10.2 High Level Hardware and Software Architecture	661
10.3 Interfaces and APIs	664
10.3.1 ECU Abstraction and its AUTOSAR Interfaces	664
10.4 Sensors/Actuators	665
10.5 I/O Hardware Abstraction	667
10.6 Complex Driver	668
11 Services	673
11.1 Overview: Generation of Service-related Model Elements	673
11.2 Extending the ECU Software Composition	676
11.3 Service Software Component Type	677
11.4 Service Proxy Component Type	680
11.5 Non Volatile Memory	682
11.5.1 Introduction	682
11.5.2 NvBlockComponent	683
11.5.3 Software-Components using <i>NVRAM data</i> of NvBlockComponents	685
11.5.4 Software-Components connected to NvBlockComponents	688
11.5.5 NvBlockDescriptor	690
11.5.5.1 Writing Strategies	692
11.5.5.2 NvBlockNeeds	699
11.5.5.3 RAM Block and ROM Block	702
11.5.5.4 NvBlockDataMapping	702
11.5.5.5 Client Server Ports	709
11.5.6 BulkNvDataDescriptor	711
11.5.7 SwcInternalBehavior of an NvBlockSwComponentType	712
12 Software Component Documentation	716
13 Service Dependencies and Service Use Cases	720
13.1 Overview	720
13.2 NvM Service Dependencies	720
13.2.1 Nvm Use Case: Permanent RAM Block	721
13.2.2 Nvm Use Case: Temporary RAM Block	722
13.2.3 Nvm Use Case: RAM Block with explicit synchronization using Mirror Interfaces	723
13.2.4 NVM Use Case: Software-Components using Nv Data provided by NvBlockSwComponentType (not ServiceSwComponent of NvM)	724
13.3 Watchdog Service Dependencies	725
13.3.1 Watchdog Service use Case: Local Supervision	725
13.3.2 Watchdog Service use Case: <i>Global Supervision Status</i> notification	727



13.3.3	Watchdog Service use Case: Control global supervision or get global supervision status	727
13.3.4	Watchdog Service use Case: Software-component wants to obtain the status of a local supervision	728
13.3.5	Watchdog Service use Case: Software-component wants to report a checkpoint	729
13.4	COM Manager Service Needs	730
13.4.1	ComM Use Case: read current ComM Mode	730
13.4.2	ComM Use Case: request ComM Mode	730
13.4.3	ComM Use Case: Software-Component acts as a Mode Manager that influences the ECU State	731
13.4.4	Read PNC ComM Mode	731
13.5	ECU State Manager Service Needs	732
13.5.1	EcuM Use Case: select Shutdown Target	732
13.5.2	EcuM Use Case: select Boot Target	733
13.5.3	EcuM Use Case: use Alarm Clock	733
13.6	BswM	734
13.6.1	Partial Networking	734
13.6.2	Mode Manager	735
13.6.3	Mode User	737
13.6.4	Mode Requester	737
13.7	Crypto Service Dependencies	738
13.7.1	Overview	738
13.7.2	Crypto Service Use Cases	740
13.7.2.1	Crypto Service Use Case: Hash calculation	740
13.7.2.2	Crypto Service Use Case: MAC calculation	741
13.7.2.3	Crypto Service Use Case: MAC verification	741
13.7.2.4	Crypto Service Use Case: generation of random numbers	742
13.7.2.5	Crypto Service Use Case: Encryption with Authenticated Encryption with Associated Data (AEAD)	742
13.7.2.6	Crypto Service Use Case: Decryption with Authenticated Encryption with Associated Data (AEAD)	743
13.7.2.7	Crypto Service Use Case: encryption	743
13.7.2.8	Crypto Service Use Case: decryption	744
13.7.2.9	Crypto Service Use Case: signature generation	745
13.7.2.10	Crypto Service Use Case: signature verification	745
13.7.2.11	Crypto Service Use Case: usage of key management	746
13.7.3	Crypto Service Job Use Cases	746
13.7.3.1	Crypto Service Use Case: usage of job API to set key valid	746
13.7.3.2	Crypto Service Use Case: usage of job API to create a random seed	747
13.7.3.3	Crypto Service Use Case: usage of job API to generate a key	747

13.7.3.4	Crypto Service Use Case: usage of job API to derive a key . . . . .	748
13.7.3.5	Crypto Service Use Case: usage of job API to execute calculation of the public value for key exchange . . . . .	748
13.7.3.6	Crypto Service Use Case: usage of job API to execute calculation of shared secret for key exchange . . . . .	749
13.7.3.7	Crypto Service Use Case: usage of job API to execute certificate parsing . . . . .	750
13.7.3.8	Crypto Service Use Case: usage of job API to execute certificate verification . . . . .	750
13.7.4	Crypto Key Management Use Cases . . . . .	751
13.7.4.1	KeyM Service Use Case: Software-Component wants check a certificate on KeyM . . . . .	751
13.7.4.2	KeyM Service Use Case: Software-Component wants to retrieve a certificate from KeyM . . . . .	752
13.7.4.3	KeyM Service Use Case: Software-Component wants to retrieve elements of a certificate from KeyM . . . . .	752
13.7.4.4	KeyM Service Use Case: Software-Component wants to check the existence of a certificate from KeyM . . . . .	753
13.7.4.5	KeyM Service Use Case: Software-Component wants to store a (derived) key in KeyM . . . . .	753
13.7.4.6	KeyM Service Use Case: Software-Component wants to store a container with (encrypted) keys in KeyM . . . . .	754
13.7.4.7	KeyM Service Use Case: Software-Component wants to verify if cryptographic operation was executed using a specific key . . . . .	754
13.8	Diagnostic Service Dependency . . . . .	755
13.8.1	Development Approach . . . . .	755
13.8.2	Function Inhibition Needs . . . . .	756
13.8.2.1	Function Inhibition Manager Service use Case: read function permission . . . . .	757
13.8.2.2	Function Inhibition Manager Use Case: react on suppressed or unavailable events . . . . .	758
13.8.3	Diagnostic Event Needs . . . . .	758
13.8.3.1	Dem Service Use Case: diagnostic monitor, debouncing by Dem . . . . .	767
13.8.3.2	Dem Service Use Case: diagnostic monitor, debouncing by SWC . . . . .	768
13.8.3.3	Dem Service Use Case: software-component provides information about operation cycles . . . . .	769
13.8.3.4	Dem Service Use Case: software-component enables reporting of DTCs in general . . . . .	769
13.8.3.5	Dem Service Use Case: software-component enables storage of subsequent DTCs . . . . .	770

13.8.3.6	Dem Service Use Case: retrieve information of the lamp status . . . . .	770
13.8.3.7	Dem Service Use Case: DEM provides information that the fault storage overflows . . . . .	771
13.8.3.8	Dem Service Use Case: software-component suppresses the storage of DTCs . . . . .	772
13.8.3.9	Dem Service Use Case: software-component informs that the PTO is active . . . . .	772
13.8.3.10	Dem Service Use Case: software-component needs information about any DTC status change . . . . .	773
13.8.3.11	Dem Service Use Case: call operation if the data of a given diagnostic event changes (I) . . . . .	773
13.8.3.12	Dem Service Use Case: call operation if the data or status of any diagnostic event changes (II) . . . . .	774
13.8.3.13	Dem Service Use Case: software-component provides data for diagnostic purposes . . . . .	775
13.8.3.14	Dem Service Use Case: software-component gets information about a specific DTC . . . . .	776
13.8.3.15	Dem Service Use Case: Software-Component wants to be triggered on Monitor Status Changes . . . . .	776
13.8.3.16	Dem Service Use Case: write parameter identifier by software-component . . . . .	777
13.8.3.17	Dem Service Use Case: read parameter identifier by software-component . . . . .	777
13.8.3.18	Dem Service Use Case: diagnostic monitor provides monitor data, debouncing by Dem . . . . .	778
13.8.3.19	Dem Service Use Case: diagnostic monitor provides monitor data, debouncing by software-component . . . . .	779
13.8.3.20	Dem Service Use Case: software-component checks whether an event is suppressed . . . . .	779
13.8.4	Diagnostic Communication Needs . . . . .	780
13.8.4.1	Dcm Service Use Case: read/write current values by Client Server Interface . . . . .	785
13.8.4.2	Dcm Service Use Case: read/write current values of specific DID by Client Server Interface . . . . .	786
13.8.4.3	Dcm Service Use Case: read/write current values by Sender Receiver Interface or Nv Data Interface . . . . .	786
13.8.4.4	Dcm Service Use Case: start/stop or request routine results . . . . .	787
13.8.4.5	Dcm Service Use Case: IO control by Client Server Interface . . . . .	788
13.8.4.6	Dcm Service Use Case: IO control by Sender Receiver Interface . . . . .	789
13.8.4.7	Dcm Service Use Case: Access to protocol, session and security information . . . . .	792

13.8.4.8	Dcm Service Use Case: Verify the access to security level . . . . .	792
13.8.4.9	Dcm Service Use Case: multiple testers access one ECU . . . . .	793
13.8.4.10	Dcm Service Use Case: Service Request Notification	793
13.8.4.11	Dcm Service Use Case: read/write and IOCtrl current values by Client Server Interface . . . . .	794
13.8.4.12	Dcm Service Use Case: A software-component acts as a "file server" to a diagnostic tester . . . . .	795
13.8.5	OBD related Needs . . . . .	796
13.8.5.1	Dem Service Use Case: In-Use-Monitor Performance Ratio calculation . . . . .	798
13.8.5.2	Dcm Service Use Case: read parameter identifier via diagnostic services by Client Server Interface . . . . .	799
13.8.5.3	Dcm Service Use Case: read parameter identifier via diagnostic services by Sender Receiver Interface . . . . .	800
13.8.5.4	Dcm Service Use Case: Request vehicle information	800
13.8.5.5	Dem Service Use Case: Read DTR data from SW-C for OBD Service \$06 . . . . .	801
13.8.5.6	Dcm Service Use Case: request control of on-board system, test or component . . . . .	802
13.8.5.7	Dem Service Use Case: In-Use-Monitoring Performance Ratio Denominator interface . . . . .	802
13.8.6	Diagnostics over IP . . . . .	803
13.8.6.1	DoIP Service Use Case: GID synchronization can be necessary if the ECU is DoIP Gid synchronization master . . . . .	806
13.8.6.2	DoIP Service Use Case: Vehicle information is broadcast or can be requested by the tester . . . . .	807
13.8.6.3	DoIP Service Use Case: Tester could also request the power status with respect to diagnostics . . . . .	807
13.8.6.4	DoIP Service Use Case: Routing activation mechanism is used which can lead to additional impact regarding authentication or confirmation . . . . .	808
13.8.6.5	DoIP Service Use Case: a DoIP entity needs to be informed when an external tester is attached or activated. . . . .	808
13.8.6.6	Service Use Case: Set and reset Warning Indicator Request bit . . . . .	809
13.8.6.7	DoIP Service Use Case: Atomic Software-Component provides the further action byte to the DoIP Service Component . . . . .	810
13.8.7	Miscellaneous Diagnostic Service Use-Cases . . . . .	811
13.8.7.1	Dcm Service Use Case: DiagnosticSessionControl . . . . .	811
13.8.7.2	Dcm Service Use Case: EcuReset . . . . .	811

13.8.7.3	Dcm Service Use Case: EcuReset ModeRapidPowerShutdown . . . . .	812
13.8.7.4	Dcm Service Use Case: CommunicationControl . . . . .	812
13.8.7.5	Dcm Service Use Case: ControlDTCSetting . . . . .	813
13.8.7.6	Dcm Service Use Case: Response On Event via diagnostic services . . . . .	813
13.8.7.7	Dcm Service Use Case: SecurityAccess . . . . .	814
13.8.7.8	Service Use Case: Atomic Software-Component implements a Hardware Shutdown . . . . .	815
13.8.7.9	Service Use Case: Upload and download of data . . . . .	815
13.9	Diagnostic Log and Trace Dependency . . . . .	816
13.9.1	Dlt use Case:Application software component transmits debug information . . . . .	817
13.10	Synchronized Time-Base Manager Dependency . . . . .	817
13.10.1	StbM Use Case: start timer and potentially get notified about its expiration . . . . .	818
13.10.2	StbM Use Case: Software-Components wants to get notifications of status changes . . . . .	819
13.10.3	StbM Use Case: Process time snapshot obtained from global time slave for diagnostics purposes . . . . .	820
13.10.4	StbM Use Case: Software-component represents a global time master . . . . .	820
13.10.5	StbM Use Case: Software-component represents a global time slave . . . . .	821
13.10.6	StbM Use Case: Software-component analyzes predictions about the time synchronization process . . . . .	822
13.11	Secure On-Board Communication . . . . .	822
13.11.1	SecOc Use Case: obtain the verification status of secure communication . . . . .	823
13.11.2	SecOc Use Case: software component retires from secure communication for a given period . . . . .	824
13.11.3	SecOc Use Case: deliver freshness to SecOC I . . . . .	824
13.11.4	SecOc Use Case: deliver freshness to SecOC II . . . . .	825
13.11.5	SecOc Use Case: deliver freshness to SecOC III . . . . .	825
13.11.6	SecOc Use Case: enable the sending of Pdus even if computation of the MAC is not possible . . . . .	826
13.11.7	SecOc Use Case: Receive notification about an authentication attempt . . . . .	826
13.12	J1939 Communication . . . . .	827
13.12.1	J1939RM Use Case: AtomicSwComponentType sends requests to the bus . . . . .	828
13.12.2	J1939RM Use Case: AtomicSwComponentType accepts requests from the bus . . . . .	829
13.12.3	J1939Dcm wants to retrieve calibration verification numbers from an application software-component . . . . .	829
13.13	Error Tracer . . . . .	830

13.13.1	Error Tracer Use Case: Default Error Tracer Service use Case: report failure . . . . .	831
13.14	Vehicle-2-X Facilities . . . . .	832
13.14.1	V2xFac Use Case: Application software component provides Vehicle specific data to the V2X-Stack for CAM transmission . . . . .	833
13.14.2	V2xFac Use Case: V2xFac notifies application software component about received messages . . . . .	833
13.14.3	V2xFac Use Case: Application software component triggers transmission of DENM message . . . . .	834
13.14.4	V2xFac Use Case: Application software component processes the <i>MAP (topology) Extended Message</i> . . . . .	835
13.14.5	V2xFac Use Case: Application software component processes Infrastructure to Vehicle Information Message . . . . .	836
13.14.6	V2xFac Use Case: Application software component processes Signal Phase And Timing Extended Message . . . . .	836
13.15	Vehicle-2-X Management . . . . .	837
13.15.1	V2xM Use Case: Application software component provides Vehicle specific data to the V2X-Stack for Position and Time information . . . . .	838
13.15.2	V2xM Use Case: Application software component needs V2X specific data from the V2X Manager . . . . .	838
13.15.3	V2xM Use Case: Application software component has soft-control over Pseudonym-Change within V2X Manager . . . . .	839
13.15.4	V2xM Use Case: Application software component has the ability to do Verification-on-Demand . . . . .	839
13.15.5	V2xM Use Case: Application software component do location based calculations . . . . .	840
13.16	Hardware Test Manager . . . . .	841
13.16.1	HtssM Service Use Case: Query results of hardware tests . . . . .	841
13.17	Intrusion Detection System Manager . . . . .	842
13.17.1	IdsM Service Use Case: AtomicSwComponentType reports security event . . . . .	842
13.17.2	IdsM Service Use Case: AtomicSwComponentType reports security event using Smart Sensor API . . . . .	843
13.17.3	IdsM Service Use Case: AtomicSwComponentType provides time stamp to IdsM . . . . .	844
14	Rapid Prototyping Scenarios . . . . .	845
14.1	Definition of Rapid Prototyping Scenario . . . . .	845
14.2	Usage of RptContainers on M1 . . . . .	848
14.3	Usage of atpSplittable for RptContainers on M1 . . . . .	850
14.4	Modifications of the Meta-Model for supporting the RPT scenario . . . . .	850
14.5	Extended Buffer Access Method . . . . .	853
14.5.1	RP Preparation . . . . .	854
14.5.2	Service Points . . . . .	858
14.5.2.1	Service Functions . . . . .	859

A	Glossary	862
B	Supported Special Use Cases	866
B.1	Asymmetric Data Transformation between a Software-Component and a Complex Driver	866
B.1.1	Overview	866
B.1.2	Modeling Aspects	867
C	History of Constraints and Specification Items	869
C.1	Constraint History of this Document according to AUTOSAR R4.0.1	869
C.1.1	Changed Constraints in R4.0.1	869
C.1.2	Added Constraints in R4.0.1	869
C.1.3	Deleted Constraints in R4.0.1	874
C.2	Constraint History of this Document according to AUTOSAR R4.0.2	874
C.2.1	Changed Constraints in R4.0.2	874
C.2.2	Added Constraints in R4.0.2	875
C.2.3	Deleted Constraints in R4.0.2	876
C.3	Constraint History of this Document according to AUTOSAR R4.0.3	876
C.3.1	Changed Constraints in R4.0.3	876
C.3.2	Added Constraints in R4.0.3	876
C.3.3	Added Specification Items in R4.0.3	879
C.3.4	Deleted Constraints in R4.0.3	894
C.3.5	Deleted Specification Items in R4.0.3	895
C.4	Constraint History of this Document according to AUTOSAR R4.1.1	895
C.4.1	Changed Constraints in R4.1.1	895
C.4.2	Added Constraints in R4.1.1	896
C.4.3	Changed Specification Items in R4.1.1	899
C.4.4	Added Specification Items in R4.1.1	900
C.4.5	Deleted Constraints in R4.1.1	903
C.4.6	Deleted Specification Items in R4.1.1	903
C.5	Constraint History of this Document according to AUTOSAR R4.1.2	904
C.5.1	Changed Constraints in R4.1.2	904
C.5.2	Added Constraints in R4.1.2	904
C.5.3	Changed Specification Items in R4.1.2	905
C.5.4	Added Specification Items in R4.1.2	905
C.5.5	Deleted Constraints in R4.1.2	906
C.5.6	Deleted Specification Items in R4.1.2	907
C.6	Constraint History of this Document according to AUTOSAR R4.1.3	907
C.6.1	Added Traceables in R4.1.3	907
C.6.2	Changed Traceables in R4.1.3	907
C.6.3	Deleted Traceables in R4.1.3	907
C.6.4	Added Constraints in R4.1.3	908
C.6.5	Changed Constraints in R4.1.3	908
C.6.6	Deleted Constraints in R4.1.3	908
C.7	Constraint History of this Document according to AUTOSAR R4.2.1	908
C.7.1	Added Traceables in R4.2.1	908

C.7.2	Changed Traceables in R4.2.1	911
C.7.3	Deleted Traceables in R4.2.1	912
C.7.4	Added Constraints in R4.2.1	912
C.7.5	Changed Constraints in R4.2.1	913
C.7.6	Deleted Constraints in R4.2.1	914
C.8	Constraint History of this Document according to AUTOSAR R4.2.2	914
C.8.1	Added Traceables in R4.2.2	914
C.8.2	Changed Traceables in R4.2.2	915
C.8.3	Deleted Traceables in R4.2.2	916
C.8.4	Added Constraints in R4.2.2	916
C.8.5	Changed Constraints in R4.2.2	917
C.8.6	Deleted Constraints in R4.2.2	918
C.9	Constraint History of this Document according to AUTOSAR R4.3.0	918
C.9.1	Added Traceables in R4.3.0	918
C.9.2	Changed Traceables in R4.3.0	921
C.9.3	Deleted Traceables in R4.3.0	923
C.9.4	Added Constraints in R4.3.0	924
C.9.5	Changed Constraints in R4.3.0	925
C.9.6	Deleted Constraints in R4.3.0	926
C.10	Constraint History of this Document according to AUTOSAR R4.3.1	926
C.10.1	Added Traceables in R4.3.1	926
C.10.2	Changed Traceables in R4.3.1	927
C.10.3	Deleted Traceables in R4.3.1	928
C.10.4	Added Constraints in R4.3.1	928
C.10.5	Changed Constraints in R4.3.1	928
C.10.6	Deleted Constraints in R4.3.1	929
C.11	Constraint History of this Document according to AUTOSAR R4.4.0	930
C.11.1	Added Traceables in R4.4.0	930
C.11.2	Changed Traceables in R4.4.0	931
C.11.3	Deleted Traceables in R4.4.0	933
C.11.4	Added Constraints in R4.4.0	934
C.11.5	Changed Constraints in R4.4.0	934
C.11.6	Deleted Constraints in R4.4.0	935
C.12	Constraint History of this Document according to AUTOSAR R19-11	936
C.12.1	Added Traceables in R19-11	936
C.12.2	Changed Traceables in R19-11	937
C.12.3	Deleted Traceables in R19-11	937
C.12.4	Added Constraints in R19-11	937
C.12.5	Changed Constraints in R19-11	939
C.12.6	Deleted Constraints in R19-11	939
C.13	Constraint History of this Document according to AUTOSAR R20-11	939
C.13.1	Added Traceables in R20-11	939
C.13.2	Changed Traceables in R20-11	940
C.13.3	Deleted Traceables in R20-11	941
C.13.4	Added Constraints in R20-11	941
C.13.5	Changed Constraints in R20-11	946



C.13.6	Deleted Constraints in R20-11	947
C.14	Constraint History of this Document according to AUTOSAR R21-11	947
C.14.1	Added Traceables in R21-11	947
C.14.2	Changed Traceables in R21-11	947
C.14.3	Deleted Traceables in R21-11	948
C.14.4	Added Constraints in R21-11	948
C.14.5	Changed Constraints in R21-11	949
C.14.6	Deleted Constraints in R21-11	966
D	Modeling of InstanceRef	968
D.1	Introduction	968
D.2	Modeling	969
D.2.1	Components and Compositions	969
D.2.2	Definition of implicit Communication Behavior	983
D.2.3	Internal Behavior	989
E	Examples	996
E.1	Examples for the Definition of variable-size Arrays	996
F	Mentioned Class Tables	1000
G	Upstream Mapping	1036
G.1	Introduction	1036
G.2	NvM	1036
G.3	Com	1044
G.4	WdgM	1060
G.5	Dcm	1061
G.6	Dem	1092
G.7	BswM	1100
G.8	MemMap	1106
G.9	RTE	1107
G.10	ECUC	1113
G.11	OS	1119
G.12	SecOC	1119
H	Splitable Elements in the Scope of this Document	1121
I	Variation Points in the Scope of this Document	1124

## References

- [1] Standardization Template  
AUTOSAR\_TPS\_StandardizationTemplate
- [2] Specification of RTE Software  
AUTOSAR\_SWS\_RTE
- [3] Virtual Functional Bus  
AUTOSAR\_EXP\_VFB
- [4] Methodology for Classic Platform  
AUTOSAR\_TR\_Methodology
- [5] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture
- [6] Basic Software Module Description Template  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate
- [7] Specification of Timing Extensions  
AUTOSAR\_TPS\_TimingExtensions
- [8] Requirements on Timing Extensions  
AUTOSAR\_RS\_TimingExtensions
- [9] Specification of ECU Resource Template  
AUTOSAR\_TPS\_ECUResourceTemplate
- [10] System Template  
AUTOSAR\_TPS\_SystemTemplate
- [11] Generic Structure Template  
AUTOSAR\_TPS\_GenericStructureTemplate
- [12] Requirements on Software Component Template  
AUTOSAR\_RS\_SoftwareComponentTemplate
- [13] Supplementary material of general blueprints for AUTOSAR  
AUTOSAR\_TR\_GeneralBlueprintsSupplement
- [14] Specification of Manifest  
AUTOSAR\_TPS\_ManifestSpecification
- [15] Information technology – Universal Coded Character Set (UCS)  
<http://www.iso.org>
- [16] Specification of I/O Hardware Abstraction  
AUTOSAR\_SWS\_IOHardwareAbstraction
- [17] ISO 17356-4: Road vehicles – Open interface for embedded automotive applications – Part 4: OSEK/VDX Communication (COM)

- [18] Specification of Module E2E Transformer  
AUTOSAR\_SWS\_E2ETransformer
- [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 Basic Software Mode Manager  
AUTOSAR\_SWS\_BSWModeManager
- [22] Specification of Communication  
AUTOSAR\_SWS\_COM
- [23] Specification of Platform Types  
AUTOSAR\_SWS\_PlatformTypes
- [24] ISO/IEC 9899:1999  
<http://www.iso.org>
- [25] AUTOSAR XML Schema Production Rules  
AUTOSAR\_TPS\_XMLSchemaProductionRules
- [26] ASAM MCD 2MC ASAP2 Interface Specification  
<http://www.asam.net>  
ASAP2-V1.51.pdf
- [27] ASAM MCD 2 Harmonized Data Objects Version 1.1  
harmonized-data-objects-V1.1.pdf
- [28] Collection of blueprints for AUTOSAR M1 models  
AUTOSAR\_MOD\_GeneralBlueprints
- [29] ISO 26262:2018 (all parts) – Road vehicles – Functional Safety  
<http://www.iso.org>
- [30] ASAM AE Calibration Data Format V2.0.0  
<http://www.asam.net>  
ASAM-AE-CDF-V2\_0\_0-Users-Guide.pdf
- [31] Specification of Operating System  
AUTOSAR\_SWS\_OS
- [32] ISO 17356-3: Road vehicles – Open interface for embedded automotive applications – Part 3: OSEK/VDX Operating System (OS)
- [33] Specification of ECU Configuration Parameters (XML)  
AUTOSAR\_MOD\_ECUConfigurationParameters
- [34] Glossary  
AUTOSAR\_TR\_Glossary
- [35] Specification of NVRAM Manager

- AUTOSAR\_SWS\_NVRAMManager
- [36] ASAM AE Functional Specification Exchange Format V1.0.0  
<http://www.asam.net>  
AE-FSX\_V1.0.0.pdf
  - [37] Specification of Watchdog Manager  
AUTOSAR\_SWS\_WatchdogManager
  - [38] Specification of ECU State Manager  
AUTOSAR\_SWS\_ECUStateManager
  - [39] Diagnostic Extract Template  
AUTOSAR\_TPS\_DiagnosticExtractTemplate
  - [40] Specification of Function Inhibition Manager  
AUTOSAR\_SWS\_FunctionInhibitionManager
  - [41] Specification of Diagnostic Event Manager  
AUTOSAR\_SWS\_DiagnosticEventManager
  - [42] Specification of Diagnostic Communication Manager  
AUTOSAR\_SWS\_DiagnosticCommunicationManager
  - [43] Road vehicles – Diagnostic communication over Internet Protocol (DoIP)  
<http://www.iso.org>
  - [44] Specification of Diagnostic Log and Trace  
AUTOSAR\_SWS\_DiagnosticLogAndTrace
  - [45] Specification of Synchronized Time-Base Manager  
AUTOSAR\_SWS\_SynchronizedTimeBaseManager
  - [46] Specification of Secure Onboard Communication  
AUTOSAR\_SWS\_SecureOnboardCommunication
  - [47] Specification of a Request Manager for SAE J1939  
AUTOSAR\_SWS\_SAEJ1939RequestManager
  - [48] Specification of Default Error Tracer  
AUTOSAR\_SWS\_DefaultErrorTracer
  - [49] Specification of Vehicle-2-X Facilities  
AUTOSAR\_SWS\_V2XFacilities
  - [50] Specification of Vehicle-2-X Management  
AUTOSAR\_SWS\_V2XManagement
  - [51] Software Process Engineering Meta-Model Specification  
<http://www.omg.org/spec/SPEM/2.0/>
  - [52] Specification of SOME/IP Transformer  
AUTOSAR\_SWS\_SOMEIPTransformer

# 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.

## 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 Archi-

ecture 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.

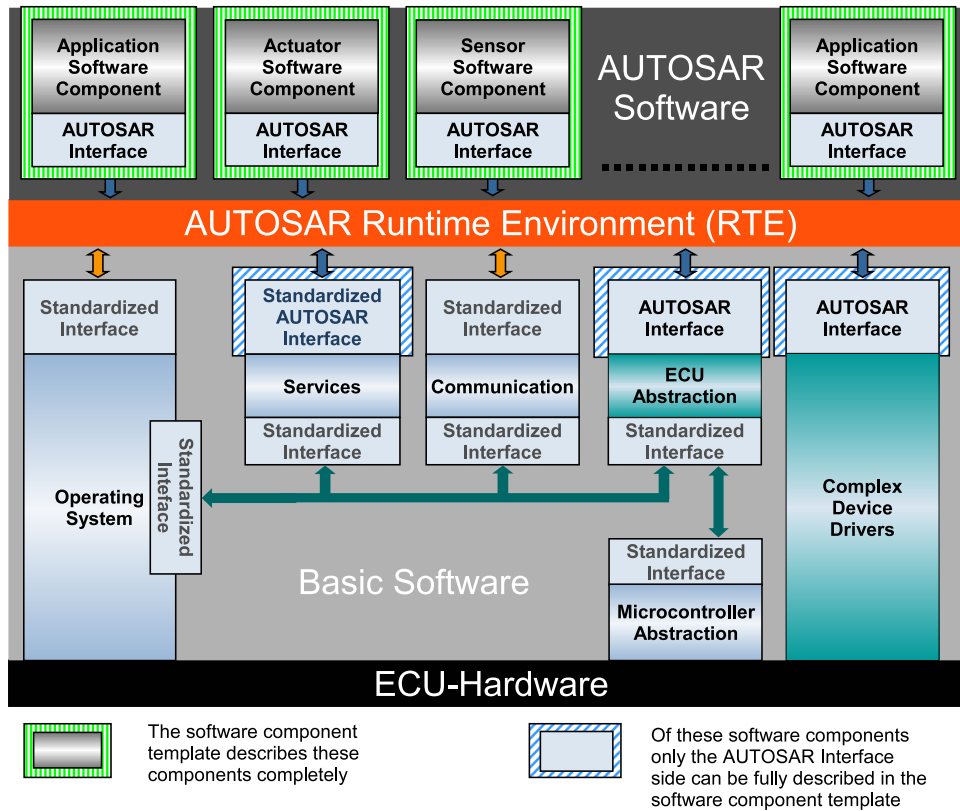


Figure 1.1: Scope of this document in the ECU SW Architecture [5]

Aspects of AUTOSAR Basic Software not relevant for the RTE are out of scope; these are covered by the Basic Software Module Description Template [6].

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 [7] as well as the corresponding requirement specification [8].

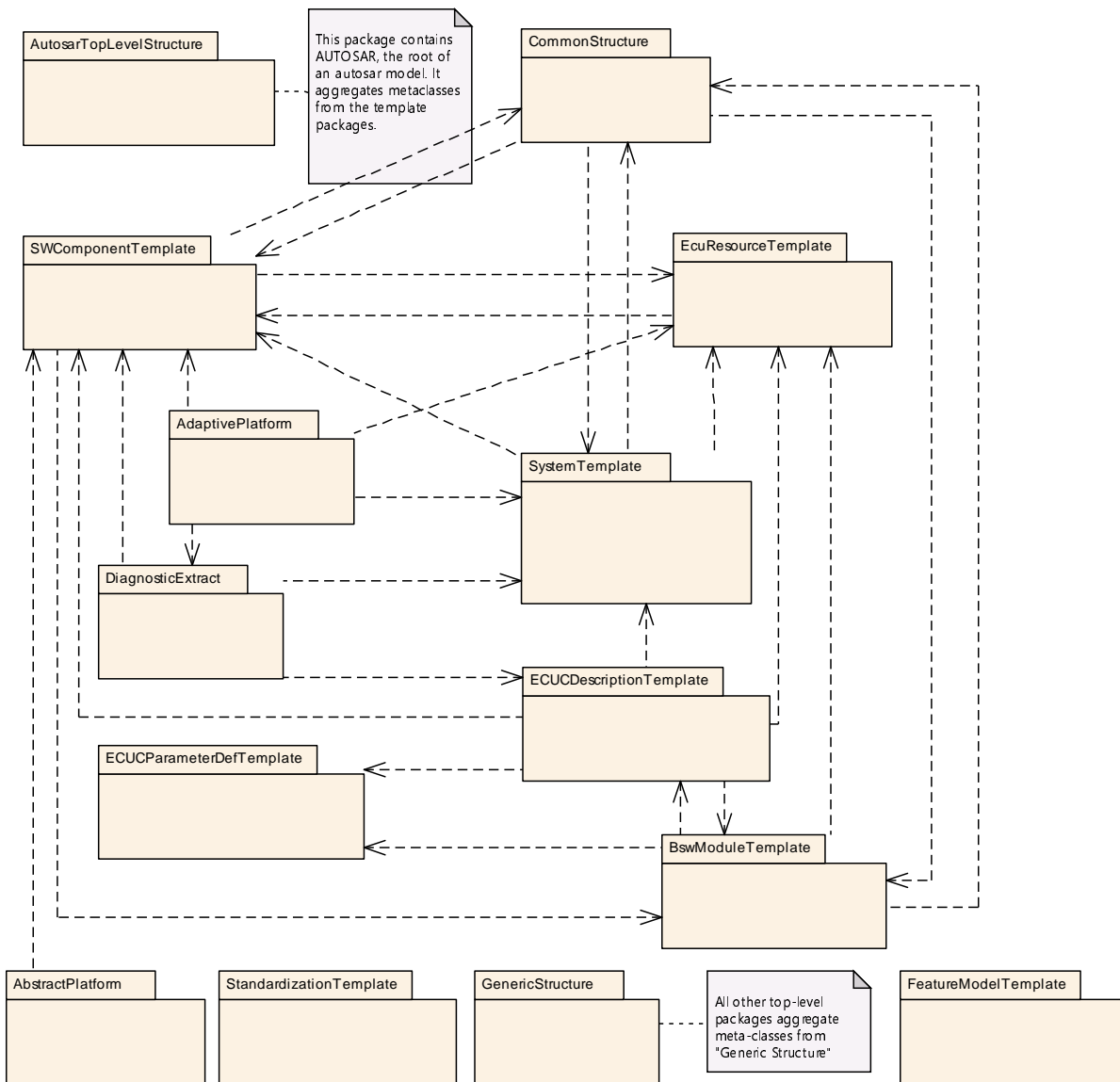
### 1.3 Organization of the Meta-Model

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 [9] and System Template [10]) 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 [11] and ECUResourceTemplate [9].

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 [11]. 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.

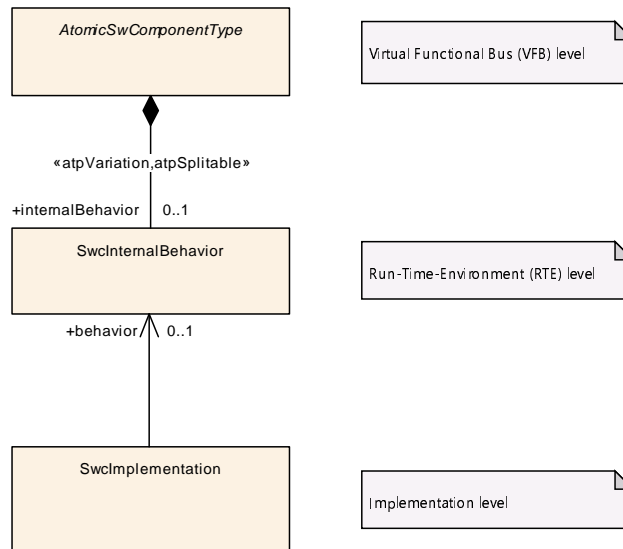


Figure 1.3: The description of a software component is done on three levels

### 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`<sup>1</sup>.

### 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`.

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



For instance, for a `ClientServerOperation` defined in the scope of a particular `ClientServerInterface` on the VFB, the behavior specifies which `RunnableEntity` is activated as a consequence of the invocation of the specific `ClientServerOperation`.

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 `<<atpSplittable>>` 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 `RunnableEntity`s 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 [6].

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.

Abbreviation	Meaning
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





<i>Abbreviation</i>	<i>Meaning</i>
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:

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

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

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

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

Let [AUTOSAR](#) be an example of a typical class table. The first rows in the table have the following meaning:

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

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

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

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

The headers in the table have the following meaning:

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

**Type:** The type of an attribute of the class.

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

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

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

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

The verbal forms for the expression of obligation specified in [TPS\_STDT\_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([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 Imposition Times of Constraints

The constraints formulated in this document have different *actual* imposition times which denote the steps in the workflow when the respective constraint has to be imposed.

Some imposition times “include” other imposition times, an example for this relation is discussed in the table 1.2.

The imposition times that are considered applicable in the scope of this document<sup>2</sup> are listed in Table 1.2.

Please note that the imposition times are intentionally rendered as technical terms such that it is possible to link back from each constraint to the definition of the affected imposition time in Table 1.2.

This document has been created to apply primarily for the *AUTOSAR classic platform* and therefore the discussed imposition times also apply exclusively to the *AUTOSAR classic platform*.

Some constraints, however, *may* also be meaningful in the context of other imposition times, applicable for other *AUTOSAR platforms*.

---

<sup>2</sup>Different imposition times may be defined in the context of other AUTOSAR standard documents

Imposition Time	Description
at the time when the contract phase generation is executed	This imposition time is aimed at the time when a software-component is ready for generating the contract phase header files such that the implementation of the software-component can be started.
at the time when the RTE is generated	<p>This imposition time denotes the step in the workflow where the model is considered complete such that the generation of the RTE can be executed.</p> <p><i>At the time when the RTE is generated, all constraints that need to be imposed at the time when the contract phase generation is executed and those that are imposed at any time in the workflow also need to be observed.</i></p> <p>In other words, a constraint that is imposed at the time when the contract phase generation is executed <b>shall also be imposed at the time when the RTE is generated.</b></p>
at any time in the workflow	<p>This means that the constraint is invariant of the imposition time and therefore universally applicable.</p> <p>Some model configurations <i>never</i> make sense and therefore need to be restricted as early as possible in order to avoid the situation where obviously non-sensical model content is unjustifiably tolerated until some step in the workflow.</p> <p>And then (considerable) effort has to be spent for cleaning up the model.</p>
at the time when the creation of the CompositionSwComponentType is finished	<p>This imposition time applies to the creation of compositions of software-components. This imposition time is considered <i>optional</i>.</p> <p>In other words, there may be use cases to deliver <code>CompositionSwComponentTypes</code> that violate constraints with this imposition time to another party.</p> <p>But it may also make sense in some cases to make sure, that a <code>CompositionSwComponentType</code> that is going to be delivered to another party fulfills the constraints associated with this binding time.</p>

**Table 1.2: Imposition Times considered in the scope of this document**

From the appearance of an imposition time that only applies to the *AUTOSAR classic platform* in the text of a constraint in this document, it **shall not be concluded that the constraint is exclusively applicable to the *AUTOSAR classic platform*.**

## 1.8 Requirements Tracing

Requirements against this document are exclusively stated in the corresponding requirements document [12].

The following table references the requirements specified in [12] 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] [TPS_SWCT_01003] [TPS_SWCT_01004] [TPS_SWCT_01668] [TPS_SWCT_01672] [TPS_SWCT_01678] [TPS_SWCT_01679] [TPS_SWCT_01694] [TPS_SWCT_01703] [TPS_SWCT_01704] [TPS_SWCT_01705] [TPS_SWCT_01716] [TPS_SWCT_01717] [TPS_SWCT_01718] [TPS_SWCT_01726] [TPS_SWCT_01728] [TPS_SWCT_01729] [TPS_SWCT_01730] [TPS_SWCT_01731] [TPS_SWCT_01732] [TPS_SWCT_01733] [TPS_SWCT_01734] [TPS_SWCT_01735] [TPS_SWCT_01740] [TPS_SWCT_01741] [TPS_SWCT_01742] [TPS_SWCT_01743] [TPS_SWCT_01745] [TPS_SWCT_01763] [TPS_SWCT_01764] [TPS_SWCT_01770] [TPS_SWCT_01776] [TPS_SWCT_01777] [TPS_SWCT_01778] [TPS_SWCT_01779] [TPS_SWCT_01780] [TPS_SWCT_01781] [TPS_SWCT_01782] [TPS_SWCT_01783] [TPS_SWCT_01784] [TPS_SWCT_01788] [TPS_SWCT_01810] [TPS_SWCT_01813] [TPS_SWCT_01814] [TPS_SWCT_01815] [TPS_SWCT_01816] [TPS_SWCT_01817] [TPS_SWCT_01818] [TPS_SWCT_01819] [TPS_SWCT_01826] [TPS_SWCT_01827] [TPS_SWCT_01829] [TPS_SWCT_01830] [TPS_SWCT_01832] [TPS_SWCT_02019] [TPS_SWCT_02020] [TPS_SWCT_02021] [TPS_SWCT_02022] [TPS_SWCT_02024] [TPS_SWCT_02025] [TPS_SWCT_02026] [TPS_SWCT_02027] [TPS_SWCT_02028] [TPS_SWCT_02031] [TPS_SWCT_02032] [TPS_SWCT_02506]
[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] [TPS_SWCT_01713] [TPS_SWCT_01714]
[RS_SWCT_00150]	AUTOSAR shall provide means to protect SW-Components from malicious SW-Components	[TPS_SWCT_01002]





Requirement	Description	Satisfied by
[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_01129] [TPS_SWCT_01132] [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_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_01690] [TPS_SWCT_01691] [TPS_SWCT_01706] [TPS_SWCT_01707] [TPS_SWCT_01708] [TPS_SWCT_01709] [TPS_SWCT_01711] [TPS_SWCT_01712] [TPS_SWCT_01715] [TPS_SWCT_01739] [TPS_SWCT_01765] [TPS_SWCT_01766] [TPS_SWCT_01767] [TPS_SWCT_01789] [TPS_SWCT_01790] [TPS_SWCT_01791] [TPS_SWCT_01808] [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] [TPS_SWCT_02505]
[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_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] [TPS_SWCT_01663]
[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_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_01000] [TPS_SWCT_01001] [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]





Requirement	Description	Satisfied by
[RS_SWCT_02030]	Specification of Communications shall be supported	[TPS_SWCT_01002] [TPS_SWCT_01025] [TPS_SWCT_01026] [TPS_SWCT_01027] [TPS_SWCT_01087] [TPS_SWCT_01106] [TPS_SWCT_01114] [TPS_SWCT_01115] [TPS_SWCT_01116] [TPS_SWCT_01117] [TPS_SWCT_01118] [TPS_SWCT_01119] [TPS_SWCT_01120] [TPS_SWCT_01121] [TPS_SWCT_01122] [TPS_SWCT_01124] [TPS_SWCT_01196] [TPS_SWCT_01198] [TPS_SWCT_01218] [TPS_SWCT_01454] [TPS_SWCT_01516] [TPS_SWCT_01517] [TPS_SWCT_01801] [TPS_SWCT_01803]
[RS_SWCT_02060]	Interaction with basic software shall be considered	[TPS_SWCT_01005] [TPS_SWCT_01043] [TPS_SWCT_01044] [TPS_SWCT_01045] [TPS_SWCT_01046] [TPS_SWCT_01556] [TPS_SWCT_01660] [TPS_SWCT_01661] [TPS_SWCT_01689] [TPS_SWCT_01693] [TPS_SWCT_01833]
[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] [TPS_SWCT_01713] [TPS_SWCT_01714]
[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] [TPS_SWCT_01736] [TPS_SWCT_01737]
[RS_SWCT_02110]	Definition of comments shall be supported	[TPS_SWCT_01062] [TPS_SWCT_01203] [TPS_SWCT_01204] [TPS_SWCT_01205] [TPS_SWCT_01206] [TPS_SWCT_01207] [TPS_SWCT_01208] [TPS_SWCT_01209] [TPS_SWCT_01211] [TPS_SWCT_01212] [TPS_SWCT_01214] [TPS_SWCT_01215] [TPS_SWCT_01216] [TPS_SWCT_01217] [TPS_SWCT_01524]
[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_01022] [TPS_SWCT_01030] [TPS_SWCT_01075] [TPS_SWCT_01098] [TPS_SWCT_01108] [TPS_SWCT_01153] [TPS_SWCT_01154] [TPS_SWCT_01155] [TPS_SWCT_01156] [TPS_SWCT_01157] [TPS_SWCT_01302] [TPS_SWCT_01303] [TPS_SWCT_01304] [TPS_SWCT_01305] [TPS_SWCT_01306] [TPS_SWCT_01307] [TPS_SWCT_01309] [TPS_SWCT_01310] [TPS_SWCT_01312] [TPS_SWCT_01313] [TPS_SWCT_01314] [TPS_SWCT_01315] [TPS_SWCT_01317] [TPS_SWCT_01318] [TPS_SWCT_01320] [TPS_SWCT_01324] [TPS_SWCT_01354] [TPS_SWCT_01355] [TPS_SWCT_01356] [TPS_SWCT_01357] [TPS_SWCT_01358] [TPS_SWCT_01359] [TPS_SWCT_01360] [TPS_SWCT_01361] [TPS_SWCT_01363] [TPS_SWCT_01366] [TPS_SWCT_01367] [TPS_SWCT_01368] [TPS_SWCT_01369] [TPS_SWCT_01469] [TPS_SWCT_01483] [TPS_SWCT_01519] [TPS_SWCT_01520] [TPS_SWCT_01521] [TPS_SWCT_01522] [TPS_SWCT_01523] [TPS_SWCT_01687] [TPS_SWCT_03500] [TPS_SWCT_03501]
[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]







Requirement	Description	Satisfied by
[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_01474] [TPS_SWCT_01475] [TPS_SWCT_01476] [TPS_SWCT_01479] [TPS_SWCT_01480] [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_01153] [TPS_SWCT_01154] [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_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] [TPS_SWCT_01664]
[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_01507] [TPS_SWCT_01515] [TPS_SWCT_01573] [TPS_SWCT_01843]
[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]





Requirement	Description	Satisfied by
[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] [TPS_SWCT_01839]
[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_01100] [TPS_SWCT_01101] [TPS_SWCT_01102] [TPS_SWCT_01103] [TPS_SWCT_01104] [TPS_SWCT_01105]
[RS_SWCT_03170]	The SW-Component template shall support fixed data exchange	[TPS_SWCT_01102] [TPS_SWCT_01103] [TPS_SWCT_01104]
[RS_SWCT_03175]	The SW-Component template shall support the definition of calibration datasets	[TPS_SWCT_01177] [TPS_SWCT_01178] [TPS_SWCT_01188] [TPS_SWCT_01793] [TPS_SWCT_01794]
[RS_SWCT_03180]	The SW-Component template shall support SAE J1939 Protocol Features	[TPS_SWCT_01076] [TPS_SWCT_01673] [TPS_SWCT_01674] [TPS_SWCT_01809]
[RS_SWCT_03181]	The SW-Component template shall support arrays of variable number of elements within the maximum size	[TPS_SWCT_01076] [TPS_SWCT_01127] [TPS_SWCT_01495] [TPS_SWCT_01601] [TPS_SWCT_01602] [TPS_SWCT_01604] [TPS_SWCT_01605] [TPS_SWCT_01606] [TPS_SWCT_01607] [TPS_SWCT_01608] [TPS_SWCT_01610] [TPS_SWCT_01612] [TPS_SWCT_01613] [TPS_SWCT_01614] [TPS_SWCT_01615] [TPS_SWCT_01617] [TPS_SWCT_01618] [TPS_SWCT_01619] [TPS_SWCT_01620] [TPS_SWCT_01621] [TPS_SWCT_01622] [TPS_SWCT_01623] [TPS_SWCT_01636] [TPS_SWCT_01641] [TPS_SWCT_01642] [TPS_SWCT_01644] [TPS_SWCT_01645] [TPS_SWCT_01647] [TPS_SWCT_01648] [TPS_SWCT_01649] [TPS_SWCT_01650] [TPS_SWCT_01793] [TPS_SWCT_01794]
[RS_SWCT_03182]	The SW-Component template shall support byte arrays of variable number of elements	[TPS_SWCT_01127]





Requirement	Description	Satisfied by
[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_01029] [TPS_SWCT_01129] [TPS_SWCT_01132] [TPS_SWCT_01134] [TPS_SWCT_01135] [TPS_SWCT_01136] [TPS_SWCT_01137] [TPS_SWCT_01138] [TPS_SWCT_01139] [TPS_SWCT_01140] [TPS_SWCT_01421] [TPS_SWCT_01422] [TPS_SWCT_01425] [TPS_SWCT_01426] [TPS_SWCT_01427] [TPS_SWCT_01453] [TPS_SWCT_01537] [TPS_SWCT_01538] [TPS_SWCT_01539] [TPS_SWCT_01540] [TPS_SWCT_01544] [TPS_SWCT_01546] [TPS_SWCT_01547] [TPS_SWCT_01577] [TPS_SWCT_01578] [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_01680] [TPS_SWCT_01690] [TPS_SWCT_01691] [TPS_SWCT_01706] [TPS_SWCT_01707] [TPS_SWCT_01708] [TPS_SWCT_01709] [TPS_SWCT_01711] [TPS_SWCT_01712] [TPS_SWCT_01715] [TPS_SWCT_01739] [TPS_SWCT_01746] [TPS_SWCT_01765] [TPS_SWCT_01766] [TPS_SWCT_01767] [TPS_SWCT_01769] [TPS_SWCT_01789] [TPS_SWCT_01790] [TPS_SWCT_01791] [TPS_SWCT_01808] [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] [TPS_SWCT_02505]
[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_01016] [TPS_SWCT_01017] [TPS_SWCT_01018] [TPS_SWCT_01019] [TPS_SWCT_01020] [TPS_SWCT_01021] [TPS_SWCT_01063] [TPS_SWCT_01064] [TPS_SWCT_01065] [TPS_SWCT_01066] [TPS_SWCT_01067] [TPS_SWCT_01071] [TPS_SWCT_01126] [TPS_SWCT_01450] [TPS_SWCT_01451] [TPS_SWCT_01552] [TPS_SWCT_01553] [TPS_SWCT_01554] [TPS_SWCT_01572] [TPS_SWCT_01581] [TPS_SWCT_01664] [TPS_SWCT_01811]
[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] [TPS_SWCT_01664]
[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] [TPS_SWCT_01768]
[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_01076] [TPS_SWCT_01078] [TPS_SWCT_01189] [TPS_SWCT_01229] [TPS_SWCT_01231] [TPS_SWCT_01235] [TPS_SWCT_01236] [TPS_SWCT_01247] [TPS_SWCT_01256] [TPS_SWCT_01295] [TPS_SWCT_01296] [TPS_SWCT_01298] [TPS_SWCT_01299] [TPS_SWCT_01300] [TPS_SWCT_01489] [TPS_SWCT_01837]





Requirement	Description	Satisfied by
[RS_SWCT_03216]	The SW-Component template shall support application data type	[TPS_SWCT_01072] [TPS_SWCT_01073] [TPS_SWCT_01179] [TPS_SWCT_01180] [TPS_SWCT_01181] [TPS_SWCT_01183] [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_01244] [TPS_SWCT_01245] [TPS_SWCT_01247] [TPS_SWCT_01249] [TPS_SWCT_01256] [TPS_SWCT_01486] [TPS_SWCT_01562] [TPS_SWCT_01564] [TPS_SWCT_01565] [TPS_SWCT_01760] [TPS_SWCT_01834] [TPS_SWCT_01835] [TPS_SWCT_01839]
[RS_SWCT_03217]	The SW-Component template shall support implementation data type	[TPS_SWCT_01006] [TPS_SWCT_01007] [TPS_SWCT_01072] [TPS_SWCT_01074] [TPS_SWCT_01183] [TPS_SWCT_01184] [TPS_SWCT_01189] [TPS_SWCT_01191] [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_01257] [TPS_SWCT_01258] [TPS_SWCT_01259] [TPS_SWCT_01442] [TPS_SWCT_01443] [TPS_SWCT_01478] [TPS_SWCT_01564] [TPS_SWCT_01565] [TPS_SWCT_01610] [TPS_SWCT_01612] [TPS_SWCT_01613] [TPS_SWCT_01614] [TPS_SWCT_01615] [TPS_SWCT_01617] [TPS_SWCT_01618] [TPS_SWCT_01619] [TPS_SWCT_01620] [TPS_SWCT_01621] [TPS_SWCT_01622] [TPS_SWCT_01647] [TPS_SWCT_01648] [TPS_SWCT_01649] [TPS_SWCT_01650] [TPS_SWCT_01700] [TPS_SWCT_01701] [TPS_SWCT_01702] [TPS_SWCT_01759] [TPS_SWCT_01772] [TPS_SWCT_01773]
[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_01594] [TPS_SWCT_01595] [TPS_SWCT_01596] [TPS_SWCT_01597] [TPS_SWCT_01598] [TPS_SWCT_01599] [TPS_SWCT_01600] [TPS_SWCT_01626] [TPS_SWCT_01812] [TPS_SWCT_03500] [TPS_SWCT_03501]
[RS_SWCT_03222]	The SW-Component template shall support error notification for transformed data communication	[TPS_SWCT_01616] [TPS_SWCT_01624] [TPS_SWCT_01626]
[RS_SWCT_03225]	The SW-Component template shall support an enhanced non-volatile (NV) memory interface	[TPS_SWCT_01141] [TPS_SWCT_01142] [TPS_SWCT_01143] [TPS_SWCT_01227] [TPS_SWCT_01228] [TPS_SWCT_01584] [TPS_SWCT_01585] [TPS_SWCT_01586] [TPS_SWCT_01587] [TPS_SWCT_01588] [TPS_SWCT_01589] [TPS_SWCT_01590] [TPS_SWCT_01662] [TPS_SWCT_01665] [TPS_SWCT_01666] [TPS_SWCT_01675] [TPS_SWCT_01754] [TPS_SWCT_01755] [TPS_SWCT_01795] [TPS_SWCT_01805] [TPS_SWCT_01806] [TPS_SWCT_01807] [TPS_SWCT_02501] [TPS_SWCT_02502] [TPS_SWCT_02503] [TPS_SWCT_02504]
[RS_SWCT_03230]	The SW-Component template shall support documentation of M1 artifacts	[TPS_SWCT_01062] [TPS_SWCT_01699]
[RS_SWCT_03240]	The SW-Component template shall support end-to-end communication protection	[TPS_SWCT_01089] [TPS_SWCT_01090] [TPS_SWCT_01091] [TPS_SWCT_01092] [TPS_SWCT_01093] [TPS_SWCT_01094] [TPS_SWCT_01095] [TPS_SWCT_01508] [TPS_SWCT_01529]





Requirement	Description	Satisfied by
[RS_SWCT_03241]	The SW-Component template shall support partial networking	[TPS_SWCT_01169] [TPS_SWCT_01170] [TPS_SWCT_01171] [TPS_SWCT_01172] [TPS_SWCT_01173] [TPS_SWCT_01174] [TPS_SWCT_01175]
[RS_SWCT_03250]	The SW-Component template shall support bidirectional communication	[TPS_SWCT_01112] [TPS_SWCT_01113] [TPS_SWCT_01454] [TPS_SWCT_01455] [TPS_SWCT_01514] [TPS_SWCT_01573]
[RS_SWCT_03260]	The SW-Component template shall support rule-based initialization of arrays	[TPS_SWCT_01484] [TPS_SWCT_01485] [TPS_SWCT_01493] [TPS_SWCT_01494] [TPS_SWCT_01495] [TPS_SWCT_01528] [TPS_SWCT_01609] [TPS_SWCT_01692]
[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_01719] [TPS_SWCT_01720] [TPS_SWCT_01721] [TPS_SWCT_01722] [TPS_SWCT_01723] [TPS_SWCT_01724] [TPS_SWCT_02046] [TPS_SWCT_02047] [TPS_SWCT_02048] [TPS_SWCT_02049] [TPS_SWCT_02050] [TPS_SWCT_02051] [TPS_SWCT_02052]
[RS_SWCT_03281]	The SW-Component template shall support post-build hooking tools for rapid prototyping	[TPS_SWCT_02047]
[RS_SWCT_03282]	The SW-Component template shall support the description of service points and rapid prototyping scenarios	[TPS_SWCT_02046] [TPS_SWCT_02047]
[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_01538] [TPS_SWCT_01539] [TPS_SWCT_01544] [TPS_SWCT_01546] [TPS_SWCT_01547] [TPS_SWCT_01746]
[RS_SWCT_03320]	The SW-Component template shall support the definition of optional elements for communication	[TPS_SWCT_01771] [TPS_SWCT_01772] [TPS_SWCT_01773] [TPS_SWCT_01774] [TPS_SWCT_01775] [TPS_SWCT_01785] [TPS_SWCT_01786] [TPS_SWCT_01821] [TPS_SWCT_01822] [TPS_SWCT_01823]

**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 an 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 similarly.

**[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](#)).
- sharing between all [SwComponentPrototypes](#) of the same [SwComponentType](#) in its [SwcInternalBehavior](#) via an aggregation of an [ParameterDataPrototype](#) in the role of [sharedParameter](#) or [constantMemory](#).

- for several `SwComponentPrototypes` (using the port-/interface-concept with `ParameterInterfaceS`).

]()

Please note:

- The definition of `perInstanceParameter` is further described in chapter 2.2.3.3.
- Chapter 2.2.3.2 provides more information about the definition of `sharedParameter` or `constantMemory`.
- For more information regarding the definition of `ParameterInterface`, please refer to chapter 2.2.3.1.

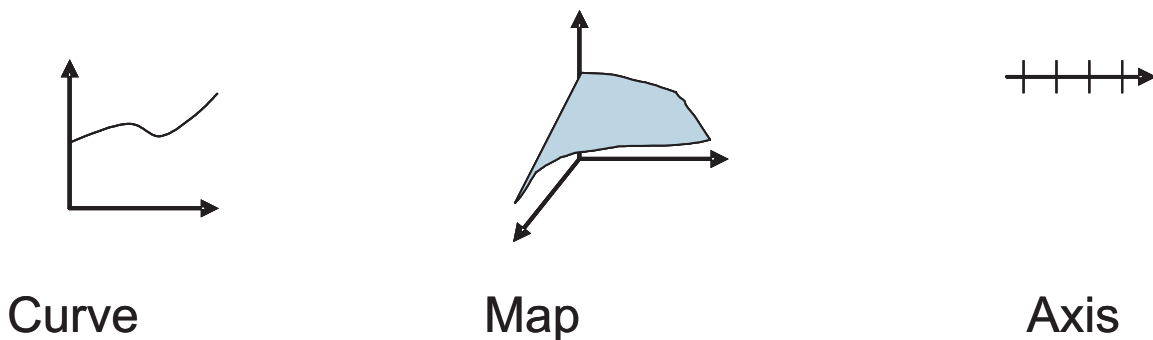


Figure 2.1: Some Categories of calibration parameters

Note: the structure of various calibration objects is visualized in [13].

## 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`.

<b>Class</b>	<code>ParameterSwComponentType</code>
<b>Package</b>	<code>M2::AUTOSARTemplates::SWComponentTemplate::Components</code>





<b>Class</b>	<b>ParameterSwComponentType</b>			
<b>Note</b>	The ParameterSwComponentType defines parameters and characteristic values accessible via provided Ports. The provided values are the same for all connected SwComponentPrototypes <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
constant Mapping	<a href="#">ConstantSpecificationMappingSet</a>	*	ref	Reference to the ConstantSpecificationMapping to be applied for the particular ParameterSwComponentType <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=constantMapping
dataType Mapping	<a href="#">DataTypeMappingSet</a>	*	ref	Reference to the DataTypeMapping to be applied for the particular ParameterSwComponentType <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=dataTypeMapping
instantiation DataDefProps	<a href="#">InstantiationDataDefProps</a>	*	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 <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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.]()

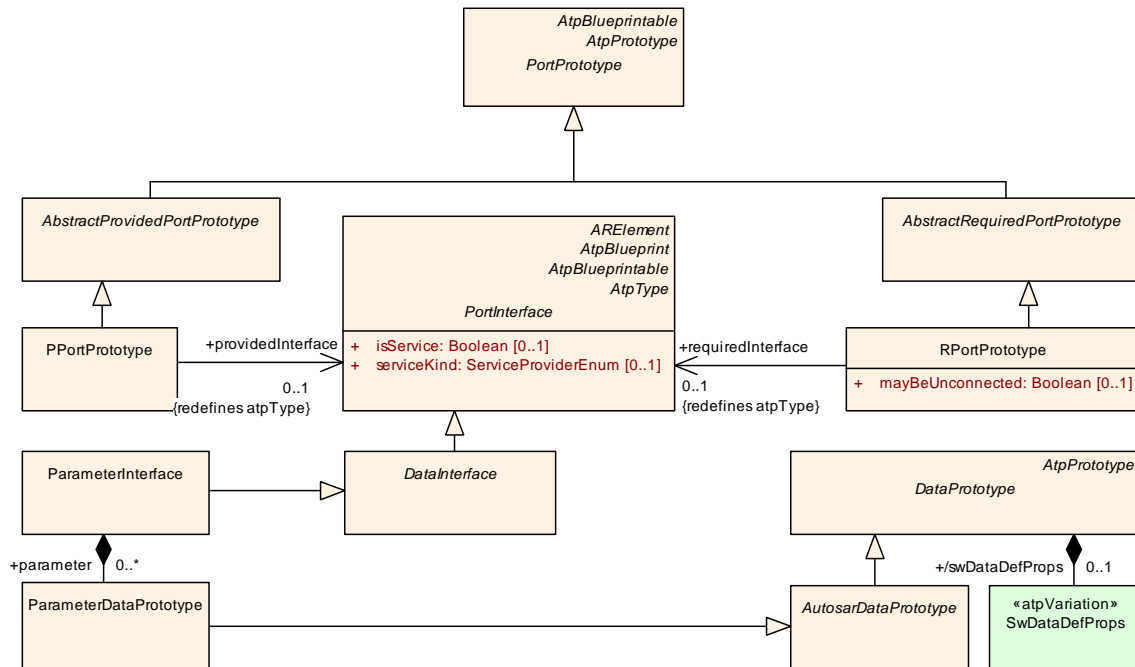
<b>Class</b>	<b>ParameterInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	A parameter interface declares a number of parameter and characteristic values to be exchanged between parameter components and software components. <b>Tags:</b> atp.recommendedPackage=PortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DataInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
parameter	<a href="#">ParameterDataPrototype</a>	*	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.](RS\_SWCT\_03190)



The compatibility rules for `ParameterInterfaces` are described in chapter 6.4; the compatibility rules for `ParameterDataPrototypes` are described in chapter 6.4.4.



**Figure 2.2: ParameterInterface**

[TPS\_SWCT\_01422] Delegation of `PortPrototypes` typed by a `ParameterInterface` [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` while the calibration system has to cope with setting the parameter synchronously on the affected ECUs.] (*RS\_SWCT\_03190*)

### 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**.]()

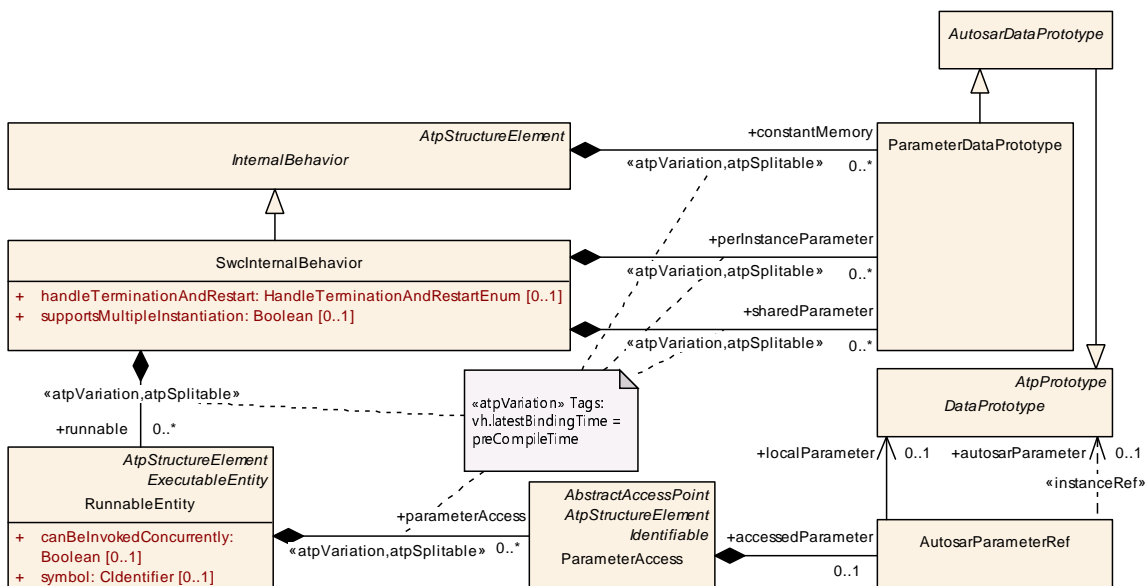


Figure 2.3: ParameterDataPrototypes in internal behavior

## 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`s 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”<sup>1</sup>.

This is technically feasible because it is always guaranteed that the `RunnableEntity`s 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 `RunnableEntity`s 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 `RunnableEntity`s they acquire the semantics of so called state-messages.

Nevertheless, directly sharing memory between `RunnableEntity`s requires a serious problem to be solved: the guarantee of data consistency among communicating `RunnableEntity`s. The `RunnableEntity`s 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.

---

<sup>1</sup>Please 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.

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

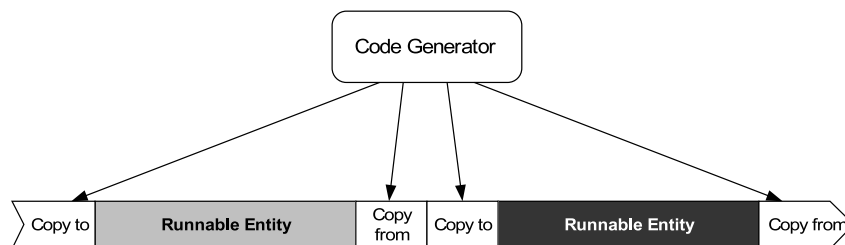
Priority ceiling allows for a non-blocking protection of shared resources. Provided that the priority scheme is static, the AUTOSAR OS is capable of temporarily raising the priority of a task that attempts to access a shared resource to the highest priority of all tasks that would ever attempt to access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



**Figure 2.4: Generation of copy routines around `RunnableEntities`**

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.

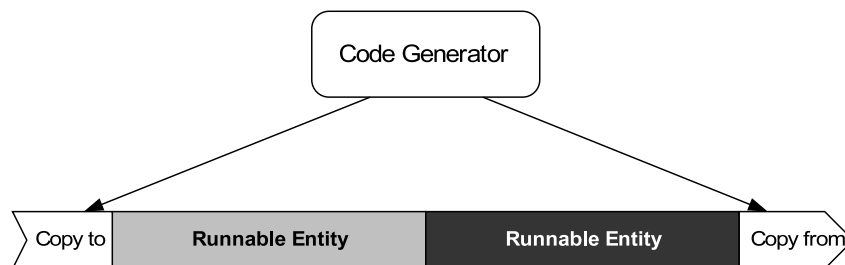
The additional copy routines as sketched in Figure 2.4 already protect the particular `RunnableEntity`s from unintended changes of concurrently accessed variables. It would, however, be possible to further optimize the process by reducing the additional code at the beginning and end of each task (see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

`AtomicSwComponentType` 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 domain, 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 of 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 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 cases<sup>2</sup>.

Admittedly, this is handled differently 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.

---

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

## 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`
- 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](#))

This approach supports adjusting the number and kind of software-component instances as well as their interconnection in a particular system variant.

**[TPS\_SWCT\_01447] Applicable binding times for model elements in the scope of the Software Component Template** [The first three cases are supporting *Post-Build* 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 [11].



## 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 owned 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.

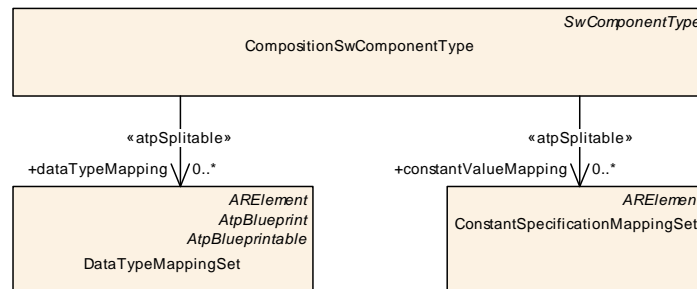
It is not required that the *initValues* of delegated PortPrototype and a PortPrototype connected by means of a DelegationSwConnector match.

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 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

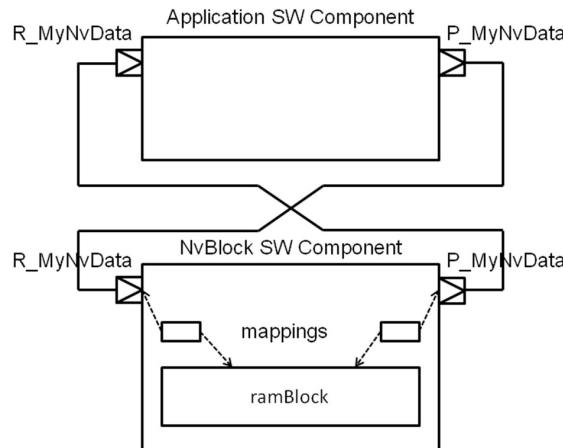
In some cases `SwComponentTypes` need to read and write the same piece of data. One of the most prominent examples for this use case is the `NvBlockSwComponentType` that factually reads and writes blocks of NvRAM.

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.

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`.

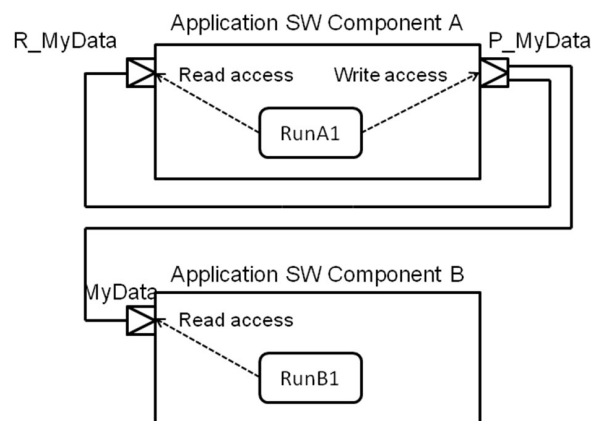


**Figure 2.7: Use Case 1 for the existence of [PRPortPrototype](#)**

### 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 [RunnableEntity](#)s 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.

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.

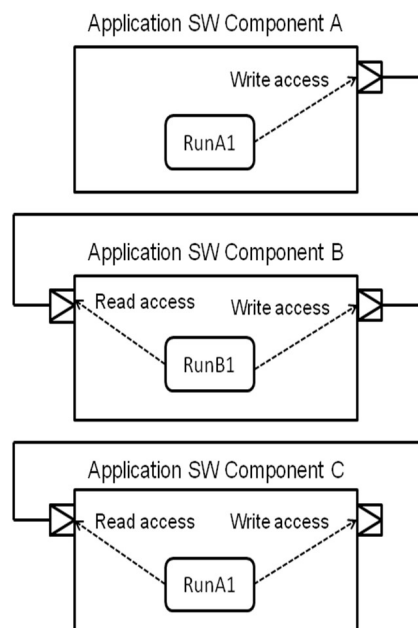


Figure 2.9: Use Case 3 for the existence of `PRPortPrototype`

### 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 and 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 3.2.2 and 4.2.1.

## 2.7 Variable-size Array Data Types

### 2.7.1 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.

#### 2.7.1.1 “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 a 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 `ApplicationArrayDataType`** [An `ApplicationArrayDataType` 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` shall 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 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 `dynamicArraySizeProfile`
- The `ImplementationDataType` aggregates a `subElement` where
  - attribute `arraySizeSemantics` exists and is set to the value `variableSize`
  - attribute `arraySizeHandling` does not exist
- One of the following conditions applies:
  - `subElement.category` is set to `VALUE` or `TYPE_REFERENCE` that eventually boils down to `VALUE` and the attribute `subElement.swDataDefProps.baseType.baseTypeDefinition.baseTypeSize` is set to the value `8` and the attribute `baseTypeEncoding` is set to `NONE`.
  - `subElement.category` is set to `TYPE_REFERENCE` and the attribute `subElement.swDataDefProps.implementationDataType` literally represents the Platform Data Type named `uint8`.
  - `subElement.category` is set to `TYPE_REFERENCE` and the attribute `subElement.swDataDefProps.implementationDataType.shortName` is set to `uint8` and `subElement.swDataDefProps.baseType.baseTypeDefinition.nativeDeclaration` does not exist.

](*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` or `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.

### 2.7.1.2 “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 `ApplicationArrayType`** [An `ApplicationArrayType` that fulfills all the following conditions shall be considered an “new-world” dynamic-size array data type.

- The `ApplicationArrayType` **defines** attribute `ApplicationArrayType.dynamicArraySizeProfile`.
- `ApplicationArrayType` 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 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 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 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**<sup>3</sup> to the maximum number of elements in the “first order” array.

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 possible value **VSA\_FULLY\_FLEXIBLE** of attribute [dynamicArraySizeProfile](#).

]([RS\\_SWCT\\_03181](#))

The described cases directly correspond to the portrayal of different kinds of variable-size arrays in Figure 2.10:

- The value [VSA\\_LINEAR](#) corresponds to the tag (a).
- The value [VSA\\_SQUARE](#) corresponds to the tag (b).
- The value [VSA\\_RECTANGULAR](#) corresponds to the tag (c).
- The value [VSA\\_FULLY\\_FLEXIBLE](#) corresponds to the tag (d).

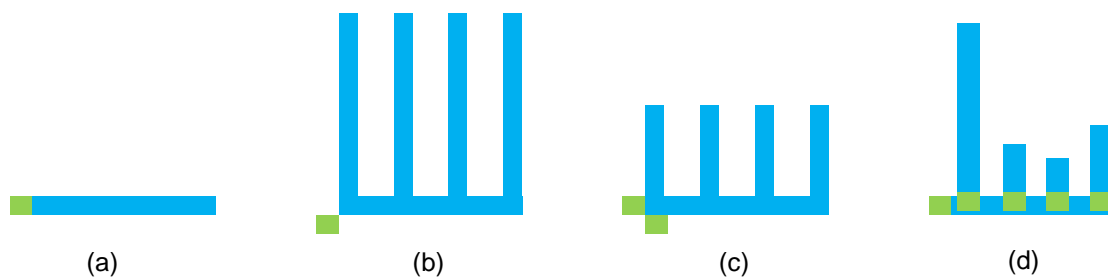
Please note that the leaf elements in a [Variable-Size Array Data Type](#) doesn't have to be primitive data types. As mentioned before, it is possible to define multiple-dimension [Variable-Size Array Data Types](#).

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

---

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





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

Please note further that the modeling of [Variable-Size Array Data Types](#) is a complex step governed by a collection of rules and constraints.

It is the expressed intent of this specification to keep the complexity of the rule set as low as possible while still providing the user with a powerful modeling framework.

The major consequence of this conclusion is to keep the modeling as straightforward as possible; in other words: intentionally cut away certain modeling variants for which acceptable workarounds within the modeling framework itself exist.

One concrete example for such a restriction is that for [ImplementationDataTypes](#), [Variable-Size Array Data Types](#) can only be defined on the level of an [AutosarDataType](#).

It is intentionally not supported to define a [Variable-Size Array Data Type](#) on the level of an [ImplementationDataTypeElement](#) because the intended semantics can be realized by assigning the value `TYPE_REFERENCE` to the [ImplementationDataTypeElement.category](#) and then let it reference to another [ImplementationDataType](#) that in turn implements the [Variable-Size Array Data Type](#).

## 2.7.2 Modeling Aspects regarding Application Data Types

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

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 of 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 [ApplicationDataType](#) 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].

### 2.7.3 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 [ImplementationDataType](#).

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 to 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 is now 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](#).

## 2.8 Optional Elements in Structures

### 2.8.1 Background

The *AUTOSAR classic platform* supports the usage of a TLV<sup>4</sup> data encoding on the SOME/IP transport layer. TLV is typically used where at least a part of the transmitted data is only *optionally* existing and filled with meaningful values.

In other words: an optional part of a data structure may exist and carry meaningful values in one instance of data transmission and be completely missing in another instance of the data transmission.

The receiving software needs to be able to identify whether the optional part exists and read its value accordingly.

The receiving software also needs to be able to still execute meaningfully if the optional part of such a data structure does not exist in the specific communication instance.

Consequently, it is necessary to be able to precisely identify the parts of a data structure that may become optional for specific instances of data transmission.

In terms of the AUTOSAR meta-model, the identification could - in principle - be attached at various levels of abstraction:

**AutosarDataType** In this case the optionality that is only needed for communication purposes would still be existing in all other usages of data types. This seems unbalanced.

Admittedly, the definition of different optionality configurations for the same data type may lead to the existence of a bunch of structurally identical data types that

---

<sup>4</sup>This abbreviation stands for tag-length-value

only vary in terms of optionality. The existence of variation points may help to mitigate this effect, though.

**PortInterface** In this case the optionality is defined where it is actually required. However, different optionality could - in principle - be defined for `DataPrototypes` typed by the same `AutosarDataType`.

This would lead to an increased effort for the definition of C data types in the context of the same `PortInterface`.

Additional constraints have been identified in the context of the definition of RTE APIs of the AUTOSAR classic platform that finally render this option as not viable.

**ComSpec** In this case (for more information please refer to section 4.5) the definition of optionality would even be more specific in comparison to the definition of optionality on the level of `PortInterfaces`.

On top of that, the task to define optionality in the vast majority of cases is done by an OEM, whereas the model definition on the level of `ComSpec` requires the existence of `SwComponentTypes` and this definition is in many cases in the domain of a supplier.

As a result of this consideration, AUTOSAR has opted for implementation of the concept of defining the optionality on the level of the `AutosarDataType`.

## 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 [11].

The type-prototype pattern is created by using references from prototype to type that are decorated with the `<<isOfType>>`.

**[constr\_10028] Existence of reference stereotyped `<<isOfType>>`** [Any reference that is decorated with the stereotype `<<isOfType>>` shall exist **at any time in the workflow.**]()

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.

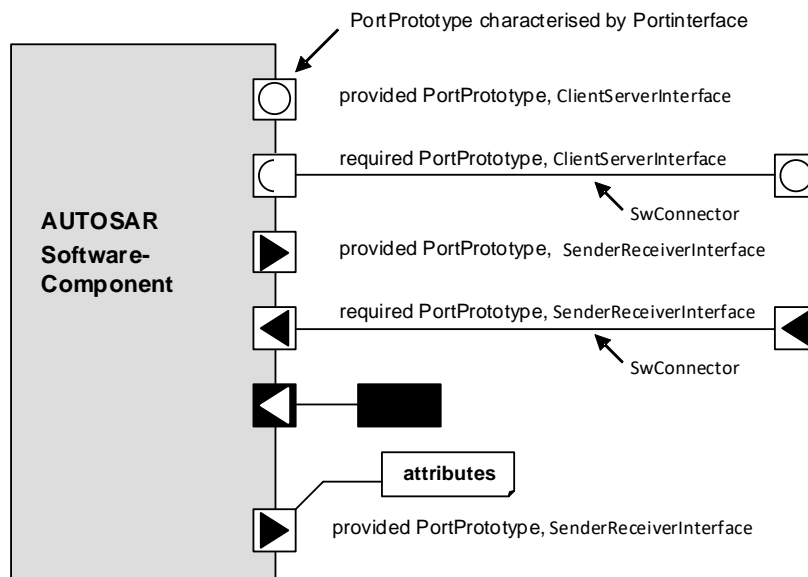
As a result, software-components can be deployed to actual ECUs at a rather late stage in the development process.

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.

<b>Class</b>	<i>SwComponentType</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Base class for AUTOSAR software components.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">AtomicSwComponentType</a> , <a href="#">CompositionSwComponentType</a> , <a href="#">ParameterSwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
consistency Needs	<a href="#">ConsistencyNeeds</a>	*	aggr	This represents the collection of ConsistencyNeeds owned by the enclosing SwComponentType. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=consistencyNeeds.shortName, consistencyNeeds.variationPoint.shortLabel vh.latestBindingTime=preCompileTime





Class	SwComponentType (abstract)			
port	<a href="#">PortPrototype</a>	*	aggr	The PortPrototypes through which this SwComponent Type can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=port.shortName, port.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portGroup	<a href="#">PortGroup</a>	*	aggr	A port group being part of this component. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
swcMapping Constraint	SwComponentMapping Constraints	*	ref	Reference to constraints that are valid for this Sw ComponentType.
swComponent Documentation	<a href="#">SwComponent Documentation</a>	0..1	aggr	This adds a documentation to the SwComponentType. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=swComponentDocumentation, swComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10
unitGroup	<a href="#">UnitGroup</a>	*	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](#))

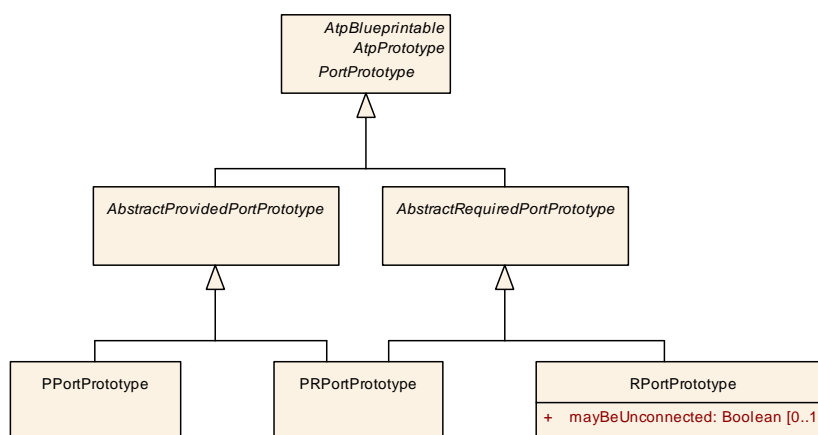
Therefore, software-components are in theory exchangeable as long as they implement the same functionality and provide the same public communication interface to the remaining system.

**[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](#).]([RS\\_SWCT\\_00010](#))

The concept of the [PortInterface](#) as another means for establishing a high degree of re-usability is described in chapter [3.4](#).

<b>Class</b>	<b>PortPrototype</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Subclasses</b>	<i>AbstractProvidedPortPrototype, AbstractRequiredPortPrototype</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
clientServer Annotation	<a href="#">ClientServerAnnotation</a>	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPort Annotation	<a href="#">DelegatedPortAnnotation</a>	0..1	aggr	Annotations on this delegated port.
ioHwAbstractionServer Annotation	<a href="#">IoHwAbstractionServerAnnotation</a>	*	aggr	Annotations on this IO Hardware Abstraction port.
logAndTraceMessageCollectionSet	<a href="#">LogAndTraceMessageCollectionSet</a>	0..1	ref	Reference to a collection of Log or Trace messages that will be used by the application. <b>Tags:</b> atp.Status=draft
modePort Annotation	<a href="#">ModePortAnnotation</a>	*	aggr	Annotations on this mode port.
nvDataPort Annotation	<a href="#">NvDataPortAnnotation</a>	*	aggr	Annotations on this non volatile data port.
parameterPort Annotation	<a href="#">ParameterPortAnnotation</a>	*	aggr	Annotations on this parameter port.
senderReceiver Annotation	<a href="#">SenderReceiverAnnotation</a>	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	<a href="#">TriggerPortAnnotation</a>	*	aggr	Annotations on this trigger port.

**Table 3.2: PortPrototype**



**Figure 3.2: Overview of PortPrototype**

[TPS\_SWCT\_01112] Semantics of [PortPrototypes](#) [[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](#))

The semantics of [PortPrototype](#) is also depicted in Figure 3.2,

**[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](#).]([RS\\_SWCT\\_03130](#), [RS\\_SWCT\\_03250](#))

Please find more information concerning the definition of “compatibility” in section 6.

<b>Class</b>	<b><a href="#">AbstractRequiredPortPrototype</a></b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	This abstract class provides the ability to become a required PortPrototype.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">PRPortPrototype</a> , <a href="#">RPortPrototype</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
requiredComSpec	<a href="#">RPortComSpec</a>	*	aggr	Required communication attributes, one for each interface element.

**Table 3.3: AbstractRequiredPortPrototype**

<b>Class</b>	<b><a href="#">AbstractProvidedPortPrototype</a></b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	This abstract class provides the ability to become a provided PortPrototype.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">PPortPrototype</a> , <a href="#">PRPortPrototype</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
providedComSpec	<a href="#">PPortComSpec</a>	*	aggr	Provided communication attributes per interface element (data element or operation).

**Table 3.4: AbstractProvidedPortPrototype**

<b>Class</b>	<b>RPortPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Component port requiring a certain port interface.			
<b>Base</b>	ARObject, <a href="#">AbstractRequiredPortPrototype</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
maybeUnconnected	Boolean	0..1	attr	If set to true, this attribute indicates that the enclosing RPortPrototype may be left unconnected and that this aspect has explicitly been considered in the software-component's design.
requiredInterface	<a href="#">PortInterface</a>	0..1	tref	The interface that this port requires. <b>Stereotypes:</b> isOfType

**Table 3.5: RPortPrototype**

<b>Class</b>	<b>PPortPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Component port providing a certain port interface.			
<b>Base</b>	ARObject, <a href="#">AbstractProvidedPortPrototype</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
providedInterface	<a href="#">PortInterface</a>	0..1	tref	The interface that this port provides. <b>Stereotypes:</b> isOfType

**Table 3.6: PPortPrototype**

<b>Class</b>	<b>PRPortPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
<b>Base</b>	ARObject, <a href="#">AbstractProvidedPortPrototype</a> , <a href="#">AbstractRequiredPortPrototype</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
providedRequiredInterface	<a href="#">PortInterface</a>	0..1	tref	This represents the PortInterface used to type the PRPortPrototype <b>Stereotypes:</b> isOfType

**Table 3.7: PRPortPrototype**

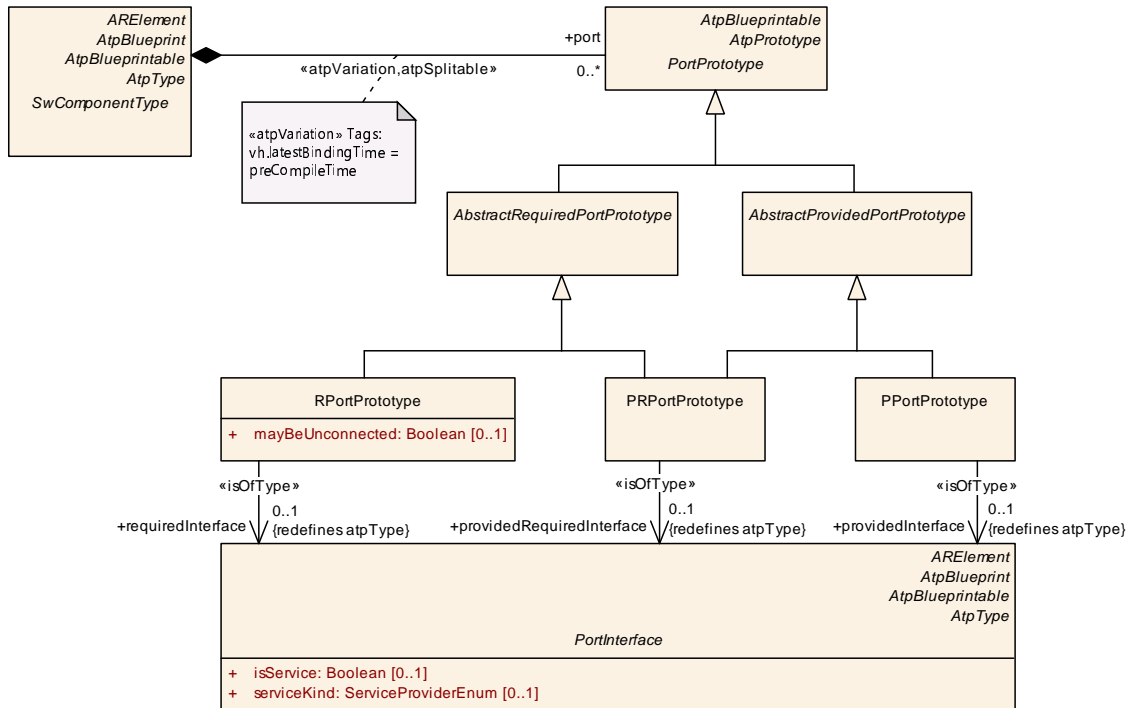
[TPS\_SWCT\_01096] **PortGroup** [[PortPrototypes](#) can be logically grouped into [PortGroups](#). This mechanism is used for implementing mode management features.] ([RS\\_SWCT\\_03201](#))

Further explanations about the semantics of meta-class [PortGroup](#) can be found in chapter 4.6.

There are cases where an [RPortPrototype](#) is intentionally left unconnected. Such a scenario would typically be reported if the RTE Generator is executed in strict mode.

By means of setting attribute [RPortPrototype.maybeUnconnected](#), the designer of the respective software-component has the ability to express that the "open" [RPortPrototype](#) is part of the final design and shall not be reported, even in strict mode.

The attribute can be used in combination of an [RPortPrototype](#) and any kind of [PortInterface](#).



**Figure 3.3: Components and Ports**

Note that the attribute applies to the entire [RPortPrototype](#), not to individual elements of the applicable [PortInterface](#).

In other words, the attribute does not affect the case, where e.g. only one [dataElement](#) of several sender/receiver [RPortPrototype](#) is left unconsidered, as described by [TPS\_SWCT\_01101].

Please note that the usage of [RPortPrototype.maybeUnconnected](#) is potentially dangerous because it removes a warning and this can be harmful if the suppression of a legitimate warning were done by mistake.

It is therefore advised to handle the existence of [RPortPrototype.maybeUnconnected](#) with care.

### 3.2.3 AtomicSwComponentType

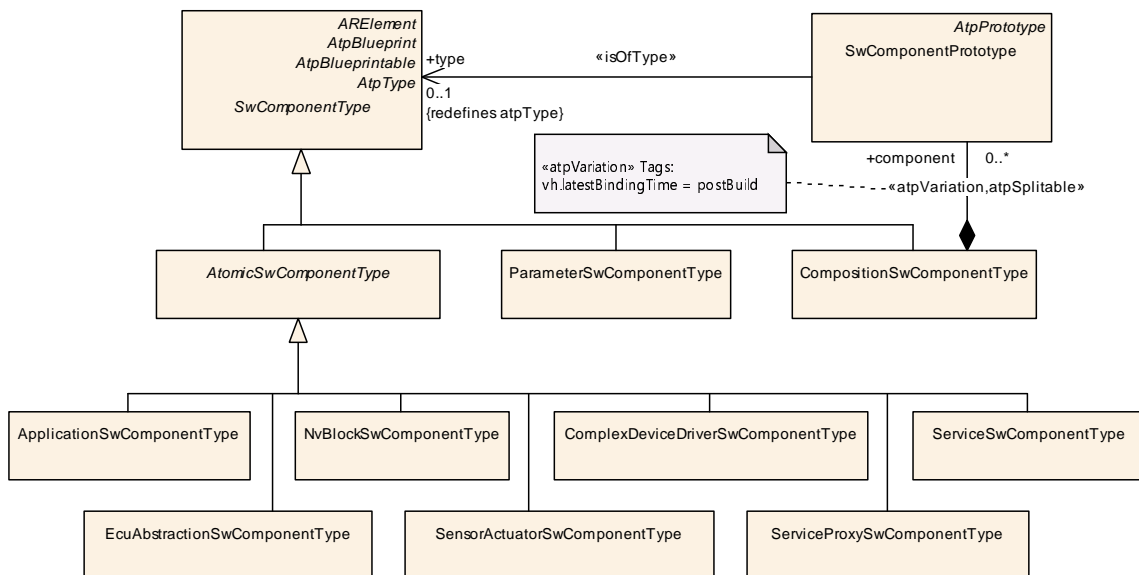
[TPS\_SWCT\_01108] Added value of an [AtomicSwComponentType](#) [As mentioned before, the term [AtomicSwComponentType](#) is a specific form of the general concept of the [SwComponentType](#). The added value of an [AtomicSwComponentType](#) is that it can aggregate an [InternalBehavior](#)] ([RS\\_SWCT\\_03040](#))

More information regarding the semantics of [InternalBehavior](#) can be found in chapter 7.

[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**.|() ]

<b>Class</b>	<b>AtomicSwComponentType</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Subclasses</b>	<a href="#">ApplicationSwComponentType</a> , <a href="#">ComplexDeviceDriverSwComponentType</a> , <a href="#">EcuAbstractionSwComponentType</a> , <a href="#">NvBlockSwComponentType</a> , <a href="#">SensorActuatorSwComponentType</a> , <a href="#">ServiceProxySwComponentType</a> , <a href="#">ServiceSwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
internalBehavior	<a href="#">SwcInternalBehavior</a>	0..1	aggr	The SwcInternalBehaviors owned by an AtomicSw ComponentType can be located in a different physical file. Therefore the aggregation is <code>&lt;&lt;atpSplitable&gt;&gt;</code> .  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=internalBehavior.shortName, internalBehavior.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	<a href="#">SymbolProps</a>	0..1	aggr	This represents the SymbolProps for the AtomicSw ComponentType.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=symbolProps.shortName

**Table 3.8: AtomicSwComponentType**



**Figure 3.4: Overview of Component Types**

There are several specialized **SwComponentTypes** to describe specific software-components used in the different parts of the AUTOSAR Layered Architecture [5]. Further details are mentioned in chapter 10 and 11.

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.

<b>Class</b>	<b>ApplicationSwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	The ApplicationSwComponentType is used to represent the application software. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtomicSwComponentType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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.

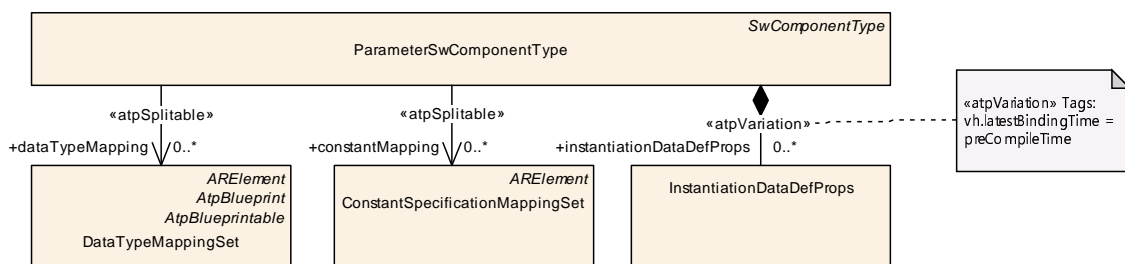


Figure 3.5: Details of `ParameterSwComponentType`

### 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`.]()

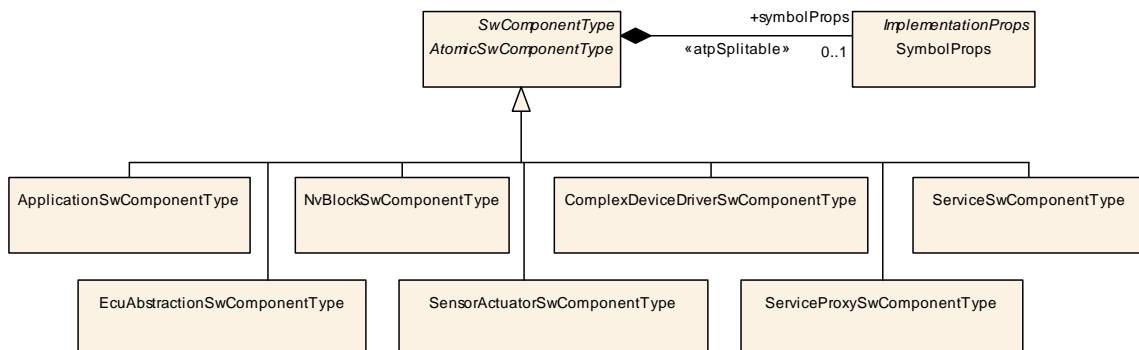
Please note that more information about the symbolic name provided by means of the attribute `symbol` of the meta-class `SymbolProps` owned by `AtomicSwComponentType` in the role `symbolProps` can be found in Figure 3.6.

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 `RunnableEntitys` [TPS\_SWCT\_01001] shall be observed.](*RS\_SWCT\_00230*)

**[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 `RunnableEntitys` owned by the `AtomicSwComponentType`.](*RS\_SWCT\_00230*)

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



**Figure 3.6: Overview of `AtomicSwComponentType`**

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.](RS\_SWCT\_00230)

### 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.

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.

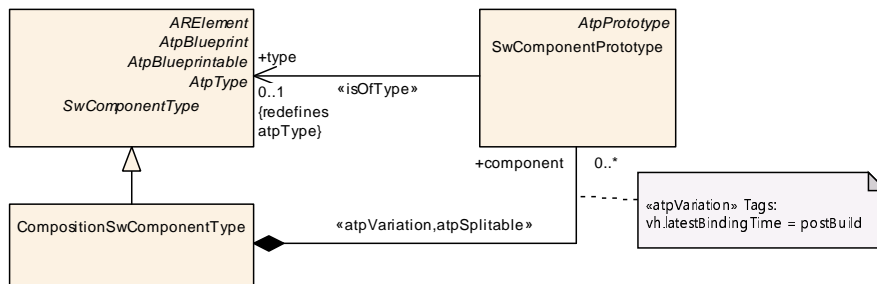


Figure 3.7: The recursive relation of software-components and compositions

**[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**.

<b>Class</b>	<b>CompositionSwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	A <b>CompositionSwComponentType</b> aggregates <b>SwComponentPrototypes</b> (that in turn are typed by <b>SwComponentTypes</b> ) as well as <b>SwConnectors</b> for primarily connecting <b>SwComponentPrototypes</b> among each others and towards the surface of the <b>CompositionSwComponentType</b> . By this means, hierarchical structures of software-components can be created. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
component	<a href="#">SwComponentPrototype</a>	*	aggr	The instantiated components that are part of this composition. The aggregation of <b>SwComponentPrototype</b> is subject to variability with the purpose to support the conditional existence of a <b>SwComponentPrototype</b> . Please be aware: if the conditional existence of <b>SwComponentPrototypes</b> is resolved post-build the deselected <b>SwComponentPrototypes</b> are still contained in the ECUs build but the instances are inactive in that they are not scheduled by the RTE.  The aggregation is marked as <b>atpSplitable</b> 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 <b>CompositionSwComponentType</b> could be to deliver an empty <b>CompositionSwComponentType</b> to e.g. a supplier for filling the internal structure. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=component.shortName, component.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	<a href="#">SwConnector</a>	*	aggr	<b>SwConnectors</b> have the principal ability to establish a connection among <b>PortPrototypes</b> . They can have many roles in the context of a <b>CompositionSwComponentType</b> . Details are refined by subclasses.  The aggregation of <b>SwConnectors</b> is subject to variability with the purpose to support variant data flow.  The aggregation is marked as <b>atpSplitable</b> in order to allow the extension of the ECU extract with <b>AssemblySw</b>







Class	CompositionSwComponentType			
				Connectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	<a href="#">ConstantSpecification MappingSet</a>	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortComSpec. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=constantValueMapping
dataType Mapping	<a href="#">DataTypeMappingSet</a>	*	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 implementors mainly because the RTE generator is not concerned about the CompositionSwComponentTypes. Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the mapping to ImplementationDataTypes is not impacting compatibility. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=dataTypeMapping
instantiation RTEEventProps	<a href="#">InstantiationRTEEvent Props</a>	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=instantiationRTEEventProps.shortLabel, instantiationRTEEventProps.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

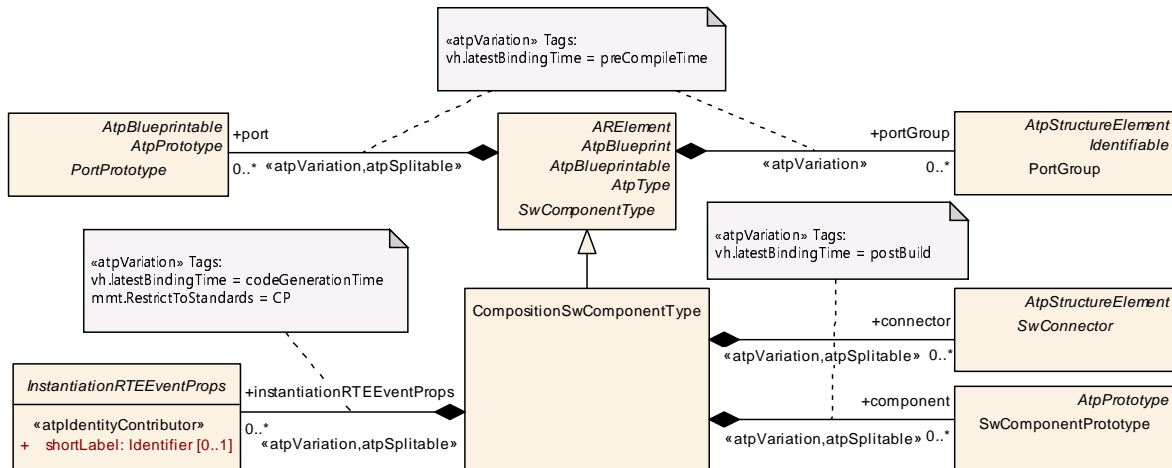
**Table 3.10: CompositionSwComponentType**

Class	SwComponentPrototype			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	Role of a software component within a composition.			
<b>Base</b>	<i>ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
type	<a href="#">SwComponentType</a>	0..1	tref	Type of the instance. <b>Stereotypes:</b> isOfType

**Table 3.11: SwComponentPrototype**

**[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](#))



**Figure 3.8: Composition and the meta-classes aggregated**

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 outside 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 for basic compatibility.
- By creating `PortPrototypes` on the surface of a specific `CompositionSwComponentType` it is explicitly decided whether the contents of an “inner” port contained in the `CompositionSwComponentType` is exposed to the outside world.

] ([RS\\_SWCT\\_03130](#))

Please note that the rules for compatibility are described in chapter 6.

Please note further 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 [10].

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 **at any time in the workflow.**]()

### 3.3.3 Connectors

**[TPS\_SWCT\_01079] `SwConnector`** [Note that `CompositionSwComponentType` also aggregates the abstract meta-class `SwConnector` for connecting the contained `SwComponentPrototypes` among each other.](*RS\_SWCT\_03130*)

More information can be found in Figure 3.8.

`CompositionSwComponentTypes` contain three 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*)

**[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*)

- `PassThroughSwConnector`, see [TPS\_SWCT\_01507].

**[constr\_1086] `SwConnector` between two specific `PortPrototypes`** [Each pair of `PortPrototypes` can only be connected by one and only one `SwConnector` **at the time when the RTE is generated.**]()

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` at any time in the workflow.]()

**[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` at any time in the workflow.]()

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.](RS\_SWCT\_03130)

**[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` at the time when the RTE is generated.]()

<b>Class</b>	<code>SwConnector</code> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition
<b>Note</b>	The base class for connectors between ports. Connectors have to be identifiable to allow references from the system constraint template.
<b>Base</b>	<code>ARObject</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code>





<b>Class</b>	<b>SwConnector</b> (abstract)			
<b>Subclasses</b>	<a href="#">AssemblySwConnector</a> , <a href="#">DelegationSwConnector</a> , <a href="#">PassThroughSwConnector</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
mapping	<a href="#">PortInterfaceMapping</a>	0..1	ref	Reference to a <a href="#">PortInterfaceMapping</a> specifying the mapping of unequal named <a href="#">PortInterface</a> elements of the two different <a href="#">PortInterface</a> s typing the two <a href="#">PortPrototypes</a> which are referenced by the <a href="#">ConnectorPrototype</a> .

**Table 3.12: SwConnector**

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](#).

<b>Class</b>	<b>AssemblySwConnector</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	AssemblySwConnectors are exclusively used to connect <a href="#">SwComponentPrototypes</a> in the context of a <a href="#">CompositionSwComponentType</a> .			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">SwConnector</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
provider	<a href="#">AbstractProvidedPortPrototype</a>	0..1	iref	Instance of providing port. <b>InstanceRef implemented by:</b> <a href="#">PPortInCompositionInstanceRef</a>
requester	<a href="#">AbstractRequiredPortPrototype</a>	0..1	iref	Instance of requiring port. <b>InstanceRef implemented by:</b> <a href="#">RPortInCompositionInstanceRef</a>

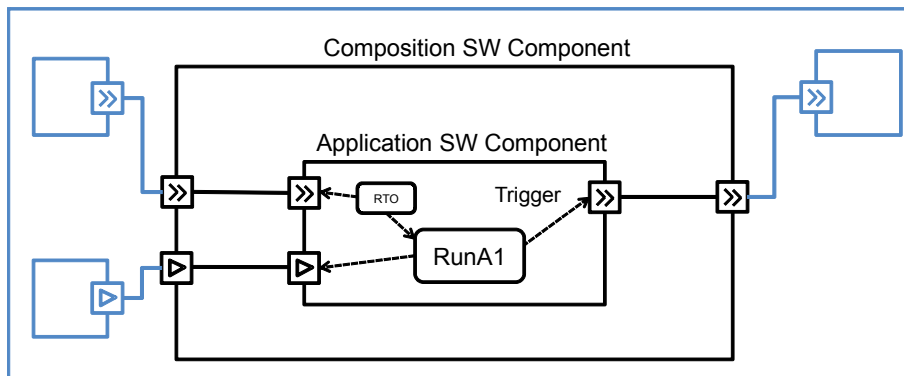
**Table 3.13: AssemblySwConnector**

<b>Class</b>	<b>DelegationSwConnector</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	A delegation connector delegates one inner <a href="#">PortPrototype</a> (a port of a component that is used inside the composition) to a outer <a href="#">PortPrototype</a> of compatible type that belongs directly to the composition (a port that is owned by the composition).			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">SwConnector</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
innerPort	<a href="#">PortPrototype</a>	0..1	iref	The port that belongs to the <a href="#">ComponentPrototype</a> in the composition <b>Tags:</b> xml.typeElement=true <b>InstanceRef implemented by:</b> <a href="#">PortInCompositionTypeInstanceRef</a>
outerPort	<a href="#">PortPrototype</a>	0..1	ref	The port that is located on the outside of the <a href="#">CompositionType</a>

**Table 3.14: DelegationSwConnector**

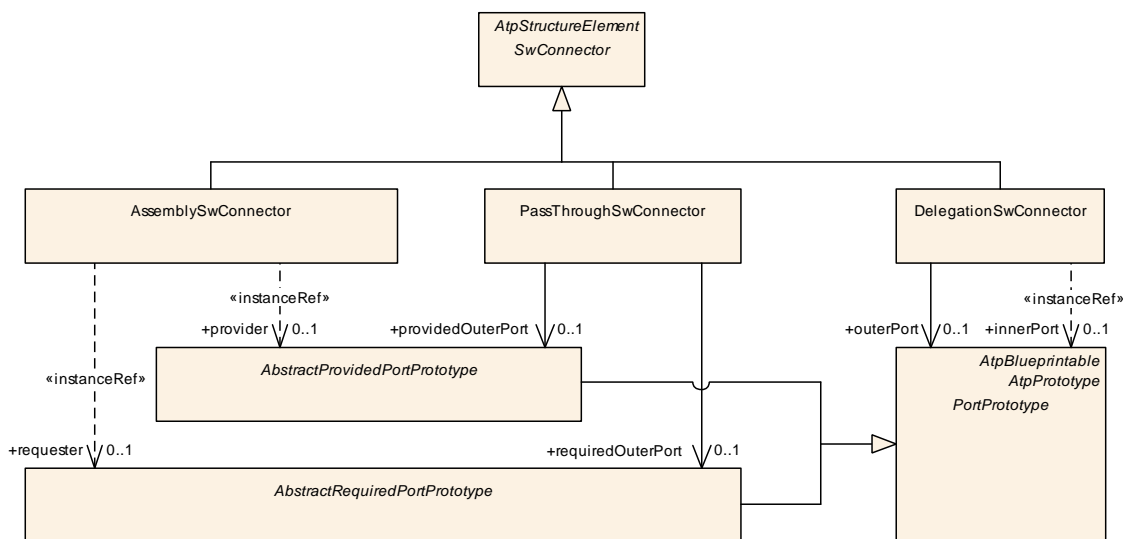
**[constr\_1860] Multiplicity of `DelegationSwConnector.innerPort`** [For each `DelegationSwConnector`, the reference `DelegationSwConnector.innerPort` shall exist **at the time when the creation of the `CompositionSwComponentType` is finished.**]()

**[constr\_1861] Multiplicity of `DelegationSwConnector.outerPort`** [For each `DelegationSwConnector`, the reference `DelegationSwConnector.outerPort` shall exist **at the time when the creation of the `CompositionSwComponentType` is finished.**]()



**Figure 3.9: Use case for `PassThroughSwConnector` (I)**

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 shove the data from (e.g. for sender-receiver communication) `RPortPrototypes` to `PPortPrototypes`.



**Figure 3.10: Connectors**

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

**[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`.] (*RS\_SWCT\_03130*)

**[constr\_1252] Creation of a loop involving a `PassThroughSwConnector` is not allowed** [at any time in the workflow, 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`.] ()

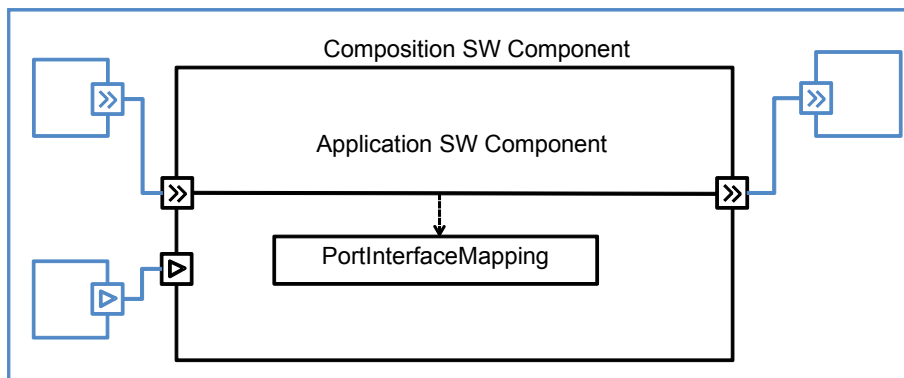


Figure 3.11: Use case for `PassThroughSwConnector` (II)

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`.

<b>Class</b>	<code>PassThroughSwConnector</code>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	This kind of <code>SwConnector</code> can be used inside a <code>CompositionSwComponentType</code> to connect two delegation <code>PortPrototypes</code> .			
<b>Base</b>	<code>ARObject</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code> , <code>SwConnector</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
providedOuterPort	<code>AbstractProvidedPortPrototype</code>	0..1	ref	This represents the provided outer delegation <code>PortPrototype</code> of the <code>PassThroughSwConnector</code> .
requiredOuterPort	<code>AbstractRequiredPortPrototype</code>	0..1	ref	This represents the required outer delegation <code>PortPrototype</code> of the <code>PassThroughSwConnector</code> .

Table 3.15: `PassThroughSwConnector`

**[constr\_1862] Multiplicity of `PassThroughSwConnector.requiredOuterPort`** [For each `PassThroughSwConnector`, the reference `PassThroughSwConnector`.

`requiredOuterPort` shall exist **at the time when the creation of the `CompositionSwComponentType` is finished.**]()

**[constr\_1863] Multiplicity of `PassThroughSwConnector.providedOuterPort`**  
[For each `PassThroughSwConnector`, the reference `PassThroughSwConnector.providedOuterPort` shall exist **at the time when the creation of the `CompositionSwComponentType` is finished.**]()

**[TPS\_SWCT\_01843] Value of `PassThroughSwConnector.category`** [Meta-class `PassThroughSwConnector` can be used in different contexts:

- The `PassThroughSwConnector` is designed to implement VFB communication and therefore is considered in the generation of the RTE. In this case the attribute `PassThroughSwConnector.category` should either not exist or should be set to the value `VFB`.
- The `PassThroughSwConnector` is used to support the early design of a signal/service translation. At some point, the early design is replaced by actual `ApplicationSwComponentTypes` that finally implement the intended communication path sketched by the early design on the VFB.

For more details about designing a signal/service translation by means of a `PassThroughSwConnector`, please refer to the TPS System Template [10]. To indicate the usage of the `PassThroughSwConnector` for the early design purpose, the value of the attribute `PassThroughSwConnector.category` should be set to `S2S_DESIGN`.

] (*RS\_SWCT\_03130*)

Please note that, on the AUTOSAR adaptive platform, the `PassThroughSwConnector` is also used for the design of a signal/service translation in a way that is very similar to the approach described in the TPS System Template [10]. More details can be found in the TPS Manifest Specification [14].

### 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*)

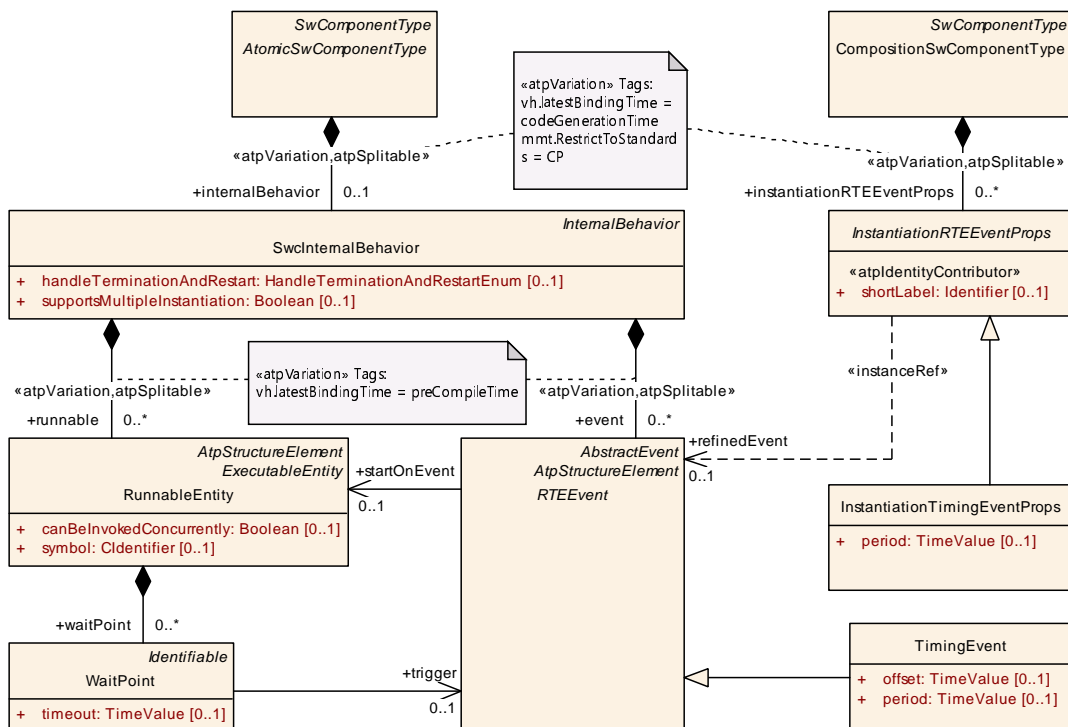
As an example for [TPS\_SWCT\_02507], please consider a software-component that implements a closed-loop control algorithm.



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.

**[constr\_1233] InstantiationTimingEventProps shall only reference TimingEvent** [at any time in the workflow, an InstantiationTimingEventProps shall only reference TimingEvent in the role refinedEvent. A reference to other kinds of RTEEvents is not supported.]()

**[constr\_1864] Multiplicity of InstantiationRTEEventProps.refinedEvent** [For each InstantiationRTEEventProps, the instance-reference InstantiationRTEEventProps.refinedEvent shall exist at the time when the RTE is generated.]()



**Figure 3.12: Instantiation specific Properties of RTEEvents**

Please note that the attribute `shortLabel` only contributes to model semantics if the ability to split the definition of the aggregation `CompositionSwComponentType.instantiationRTEEventProps` over several physical files is actually utilized<sup>1</sup>.

More explanation about the ability to split models over physical files can be found in [11].

<sup>1</sup>In which case the `shortLabel` serves as a part of the splitkey

<b>Class</b>	<b>InstantiationRTEEventProps</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	This meta-class represents the ability to refine the properties of RTEEvents for particular instances of a software component.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	InstantiationTimingEventProps			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
refinedEvent	RTEEvent	0..1	iref	This instance ref denotes the Timing Event for which the period shall be refined on an instance level. <b>InstanceRef implemented by:</b> InstanceEventInCompositionInstanceRef
shortLabel	Identifier	0..1	attr	The main purpose of the shortLabel is to contribute to the splitkey of aggregations that are <<atpSplittable>>. <b>Stereotypes:</b> atpIdentityContributor

**Table 3.16: InstantiationRTEEventProps**

### 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](#) are used to support a design-by-contract work-flow, 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](#) (see Figure 3.14) represent a pivotal point in the AUTOSAR concept.

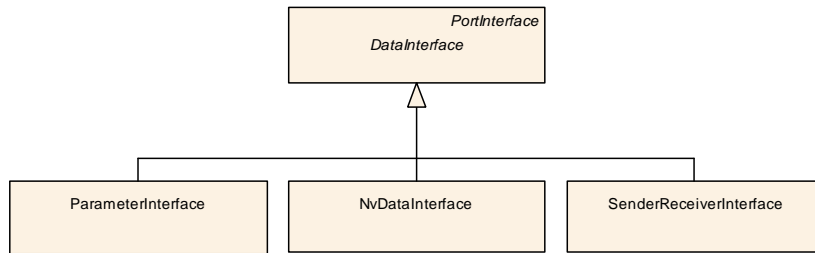
Please note that a [PortInterface](#) creates a name space for the information contained. This allows for defining the details of a specific [PortInterface](#) without having to care for possible side effects on other [PortInterfaces](#). Again, this property of the AUTOSAR concept directly supports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



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

[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](#))

Please find more details about the specialization of the [PortInterface](#) concept in chapter [4.2.3](#) and [4.2.2](#).

<b>Class</b>	<b>PortInterface</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Abstract base class for an interface that is either provided or required by a port of a software component.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">ClientServerInterface</a> , <a href="#">DataInterface</a> , <a href="#">ModeSwitchInterface</a> , <a href="#">TriggerInterface</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
isService	Boolean	0..1	attr	This flag is set if the PortInterface is to be used for communication between an <ul style="list-style-type: none"> <li>• <a href="#">ApplicationSwComponentType</a> or</li> <li>• <a href="#">ServiceProxySwComponentType</a> or</li> <li>• <a href="#">SensorActuatorSwComponentType</a> or</li> <li>• <a href="#">ComplexDeviceDriverSwComponentType</a></li> <li>• <a href="#">ServiceSwComponentType</a></li> <li>• <a href="#">EcuAbstractionSwComponentType</a></li> </ul> and a <a href="#">ServiceSwComponentType</a> (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set.
serviceKind	<a href="#">ServiceProviderEnum</a>	0..1	attr	This attribute provides further details about the nature of the applied service.

**Table 3.17: PortInterface**

[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*)

<b>Class</b>	<i>DataInterface</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	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).			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>PortInterface</i> , <i>Referrable</i>			
<b>Subclasses</b>	<i>NvDataInterface</i> , <i>ParameterInterface</i> , <i>SenderReceiverInterface</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

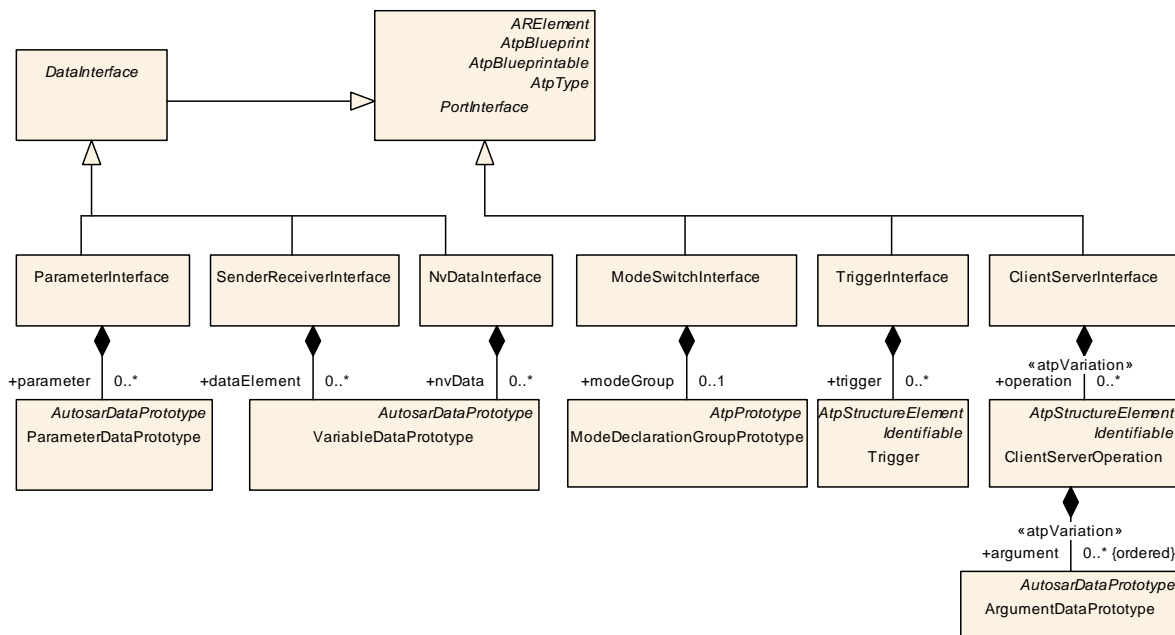
**Table 3.18: DataInterface**

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.

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 **at the time when the RTE is generated**.

Subclasses of `DataInterface` make an exception to this rule and can be used for creating connections to each other.] ()



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

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 `PortPrototype` typed by a `ParameterInterface` can **only** be owned by a `ParameterSwComponentType` or a `CompositionSwComponentType` **at any time in the workflow.**]  
( )

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 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.

The creation of an `AssemblySwConnector` between `PortPrototypes` where the respective `PortInterfaces` have set attribute `isService` to `true` puts requirements on the nature of the enclosing `SwComponentTypes`.

For example, it does not make sense to establish a service communication between two `SwComponentPrototypes` that each are typed by `ApplicationSwComponentTypes`. For service communication, at least one of the participating `SwComponentPrototypes` has to be typed by a `ServiceSwComponentType`.

**[constr\_10067] Creation of `AssemblySwConnector` for service communication** [If an `AssemblySwConnector` is created between two `PortPrototypes` and the affected `PortInterfaces` set the attribute `isService` to the value `true`, then at least one of the `SwComponentPrototypes` shall be typed by a `ServiceSwComponentType`. This constraint shall be imposed **at the time when the RTE is generated.**]  
( )

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).] (*RS\_SWCT\_00030*)

[TPS\_SWCT\_01004] **Specific 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.] (*RS\_SWCT\_00030*)

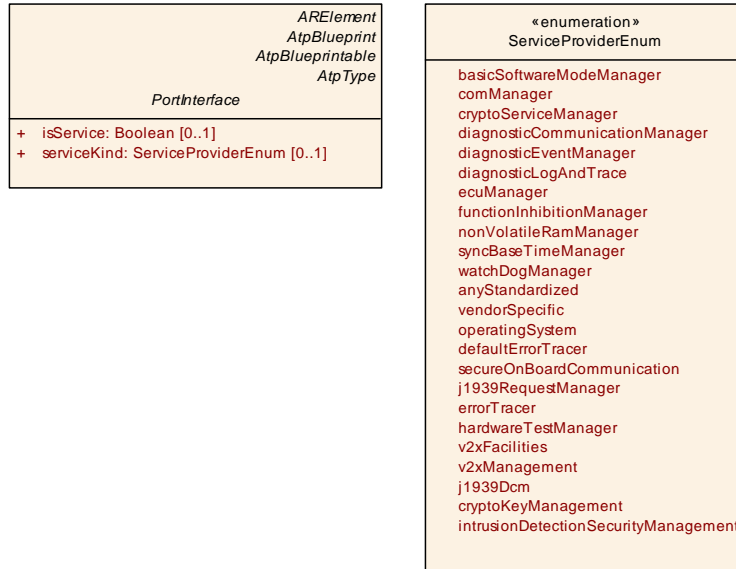


Figure 3.15: `PortInterfaces` and AUTOSAR services

[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` at any time in the workflow.] ()

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 <code>ServiceProviderEnum</code> <b>Tags:</b> atp.EnumerationLiteralIndex=0
basicSoftwareModeManager	The service relates to the Basic Software Mode Manager (BswM) <b>Tags:</b> atp.EnumerationLiteralIndex=1
comManager	The service relates to the COM Manager (ComM). <b>Tags:</b> atp.EnumerationLiteralIndex=2
cryptoKeyManagement	The service relates to the Key Manager (KeyM). <b>Tags:</b> atp.EnumerationLiteralIndex=23
cryptoServiceManager	The service relates to the Crypto Service Manager (CsM). <b>Tags:</b> atp.EnumerationLiteralIndex=3
defaultErrorTracer	The service relates to the Default Error Tracer (DET) <b>Tags:</b> atp.EnumerationLiteralIndex=4
diagnosticCommunicationManager	The service relates to the Diagnostic Communication Manager (DCM). <b>Tags:</b> atp.EnumerationLiteralIndex=6





<b>Enumeration</b>	<b>ServiceProviderEnum</b>
diagnosticEvent Manager	The service relates to the Diagnostic Event Manager (DEM). <b>Tags:</b> atp.EnumerationLiteralIndex=7
diagnosticLogAnd Trace	The service relates to the Diagnostic Log and Trace (DLT). <b>Tags:</b> atp.EnumerationLiteralIndex=8
ecuManager	The service relates to the ECU Manager (EcuM). <b>Tags:</b> atp.EnumerationLiteralIndex=9
errorTracer	This service relates to the error tracer. <b>Tags:</b> atp.EnumerationLiteralIndex=18
functionInhibition Manager	The service relates to the Function Inhibition Manager (FIM). <b>Tags:</b> atp.EnumerationLiteralIndex=10
hardwareTest Manager	This service relates to the hardware test manager. <b>Tags:</b> atp.EnumerationLiteralIndex=19
intrusionDetection Security Management	The service relates to the intrusion detection security management (IdsM). <b>Tags:</b> atp.EnumerationLiteralIndex=24
j1939Dcm	This service relates to the J1939 Dcm. <b>Tags:</b> atp.EnumerationLiteralIndex=22
j1939Request Manager	The service relates to the J1939Rm. <b>Tags:</b> atp.EnumerationLiteralIndex=11
nonVolatileRam Manager	The service relates to the Non-Volatile RAM Manager (NvM). <b>Tags:</b> atp.EnumerationLiteralIndex=12
operatingSystem	The service relates to the Operating System (OS). <b>Tags:</b> atp.EnumerationLiteralIndex=13
secureOnBoard Communication	The service relates to the SecOc module. <b>Tags:</b> atp.EnumerationLiteralIndex=14
syncBaseTime Manager	The service relates to the Sync Time Base Manager (StbM). <b>Tags:</b> atp.EnumerationLiteralIndex=15
v2xFacilities	This service relates to the Vehicle to X facilities. <b>Tags:</b> atp.EnumerationLiteralIndex=20
v2xManagement	This service relates to the Vehicle to X management. <b>Tags:</b> atp.EnumerationLiteralIndex=21
vendorSpecific	This value denotes a vendor-specific service. <b>Tags:</b> atp.EnumerationLiteralIndex=16
watchDogManager	The service relates to the Watchdog Manager (WdgM). <b>Tags:</b> atp.EnumerationLiteralIndex=17

**Table 3.19: ServiceProviderEnum**

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 relevant for communication are attached to [PortPrototypes](#).] ([RS\\_SWCT\\_00010](#), [RS\\_SWCT\\_00080](#), [RS\\_SWCT\\_00110](#), [RS\\_SWCT\\_02030](#), [RS\\_SWCT\\_03010](#))

Please note that the dynamic attributes relevant for communication are described in chapter [4.5](#).

### 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 [PortInterface](#) are:

- 2C: Two's complement
- IEEE754: floating-point numbers
- ISO-8859-1: single-byte coded character
- ISO-8859-2: single-byte coded character
- WINDOWS-1252: single-byte coded character
- UTF-8: UCS Transformation Format 8
- UTF-16: Character encoding for Unicode *code points* based on 16 bit *code units* [15]
- UCS-2: Universal Character Set 2



- NONE: Unsigned Integer
- BOOLEAN: This represents an integer to be interpreted as boolean.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1046] Applicability of [constr\_1045]** [[constr\_1045] applies **at the time when the contract phase generation is executed 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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

Note: [constr\_1295] corresponds to [SWS\_RTE\_07670].

## 4.2.2 Sender Receiver Communication

### 4.2.2.1 Sender Receiver Interface

**[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.]([RS\\_SWCT\\_02030](#))

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





<b>Class</b>	<b>SenderReceiverInterface</b>			
metaDataItemSet	<a href="#">MetaDatumSet</a>	*	aggr	This aggregation defines fixed sets of meta-data items associated with dataElements of the enclosing SenderReceiverInterface

**Table 4.1: SenderReceiverInterface**

<b>Class</b>	<b>InvalidationPolicy</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataElement	<a href="#">VariableDataPrototype</a>	0..1	ref	Reference to the dataElement for which the InvalidationPolicy applies.
handleInvalid	<a href="#">HandleInvalidEnum</a>	0..1	attr	This attribute controls how invalidation is applied to the dataElement.

**Table 4.2: InvalidationPolicy**

<b>Enumeration</b>	<b>HandleInvalidEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication
<b>Note</b>	Strategies of handling the reception of invalidValue.
<b>Literal</b>	<b>Description</b>
dontInvalidate	Invalidation is switched off. <b>Tags:</b> atp.EnumerationLiteralIndex=0
externalReplacement	Replace a received invalidValue. The replacement value is sourced from the externalReplacement. <b>Tags:</b> atp.EnumerationLiteralIndex=1
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. <b>Tags:</b> atp.EnumerationLiteralIndex=2
replace	Replace a received invalidValue. The replacement value is specified by the initialValue. <b>Tags:</b> atp.EnumerationLiteralIndex=3

**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 atomic<sup>1</sup> 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.] ([RS\\_SWCT\\_02030](#))

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

Further information about the related concept of an `invalidValue` is provided in chapter 5.4.2

**[constr\_1865] Existence of `InvalidationPolicy.dataElement`** [For each `InvalidationPolicy`, the reference `InvalidationPolicy.dataElement` shall exist at the time when the contract phase generation is executed.]()

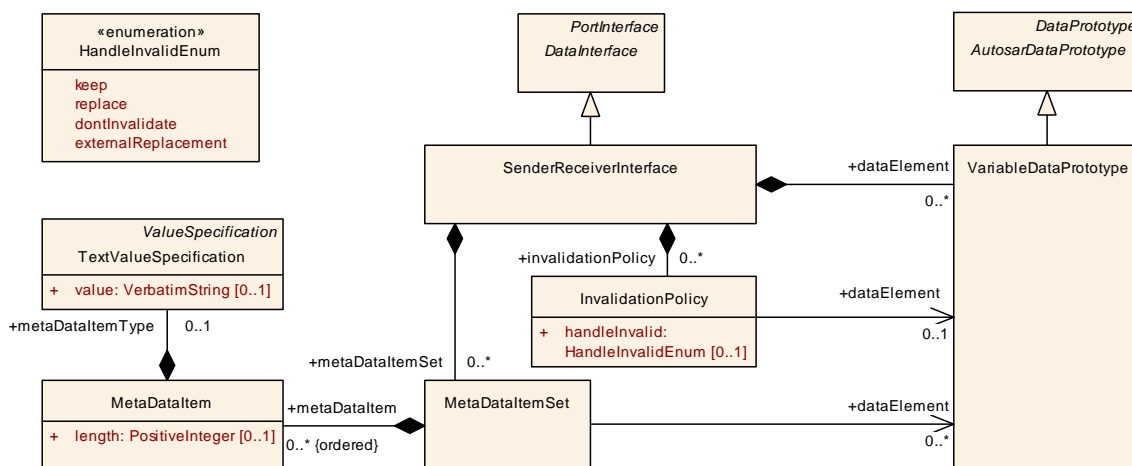


Figure 4.1: `dataElements` of a `SenderReceiverInterface`

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 [11].

**[TPS\_SWCT\_01116] `swImplPolicy`** [The `swImplPolicy` 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.] (*RS\_SWCT\_02030*)

Please note that the `swImplPolicy` is described in section 5.4.

**[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` at any time in the workflow.]()

**[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 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 [10].

**[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 an n:1 (i.e. many senders, one receiver) communication pattern.] ([RS\\_SWCT\\_02030](#))

**[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 **at any time in the workflow**.] ()

Factually, [\[constr\\_1033\]](#) is not applicable to a scenario where several [PRPortPrototypes](#) are connected by a chain of [AssemblySwConnectors](#) or [PassThroughSwConnectors](#).

	<a href="#">RPortPrototype</a>	<a href="#">PPortPrototype</a>	<a href="#">PRPortPrototype</a>
<a href="#">RPortPrototype</a>	No	Yes	Yes
<a href="#">PPortPrototype</a>	Yes	No	Yes
<a href="#">PRPortPrototype</a>	Yes	Yes	Yes

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

**[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 **at any time in the workflow**.] ()

<a href="#">innerPort</a>	<a href="#">outerPort</a>		
	<a href="#">RPortPrototype</a>	<a href="#">PPortPrototype</a>	<a href="#">PRPortPrototype</a>
<a href="#">RPortPrototype</a>	Yes	No	Yes
<a href="#">PPortPrototype</a>	No	Yes	Yes
<a href="#">PRPortPrototype</a>	Yes	Yes	Yes

**Table 4.5: Supported connections by [DelegationSwConnector](#) 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 **at any time in the workflow**.] ()

#### 4.2.2.2 Meta-data on the Application Software Level

There are cases where information available on different levels in the AUTOSAR basic software stack need to be made available as meta-data on the application layer in order to make the overall software function properly.

One example could be a software-component that is involved with communication using the J1939 protocol.

In such a case, the semantics of the information transmitted is strongly bound to the source address and the sender needs to be able to set the source address individually.

**[TPS\_SWCT\_01801] Support for Meta-Data** [Meta-data on the application software level can only be made available in the context of a [SenderReceiverInterface](#). No other kind of [PortInterface](#) supports the definition of meta-data.] ([RS\\_SWCT\\_02030](#))

**[TPS\_SWCT\_01802] Definition of meta-data in the context of a [SenderReceiverInterface](#)** [The definition of meta-data in the context of a [SenderReceiverInterface](#) involves two aspects:

- The available meta-data are defined by means of an ordered aggregation of meta-class [MetaDataItem](#) at [MetaDataItemSet](#) that in turn is aggregated in the role [SenderReceiverInterface.metaDataItemSet](#).
- The involvement of [dataElements](#) with meta-data is specified by means of the reference [MetaDataItemSet.dataElement](#). In other words, the [dataElements](#) that are referenced by a [MetaDataItemSet](#) in the role [dataElement](#) are involved with meta-data handling.

]()

<b>Class</b>	<b>MetaDatum</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	This meta-class represents a single meta-data item.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
length	PositiveInteger	0..1	attr	This attribute determines the length of the MetaDatum at run-time.
metaDatumType	<a href="#">TextValueSpecification</a>	0..1	aggr	This aggregation contributes the specification of the concrete meta-data item type.

**Table 4.6: MetaDatum**

<b>Class</b>	<b>MetaDataItemSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	This meta-class represents the ability to define a set of meta-data items to be used in SenderReceiver Interfaces.			
<b>Base</b>	<i>ARObject</i>			





Class	MetaDatumSet			
Attribute	Type	Mult.	Kind	Note
dataElement	<a href="#">VariableDataPrototype</a>	*	ref	This reference identifies the dataElement for which the ordered list of meta-data items is defined.
metaDatumItem (ordered)	<a href="#">MetaDatumItem</a>	*	aggr	This aggregation represents the ordered definition of meta-data items.

**Table 4.7: MetaDatumSet**

**[constr\_1726] Ordering of [MetaDatumItemSet.metaDatumItem](#)** [The ordering of the elements of [MetaDatumItemSet.metaDatumItem](#) shall be done such that the [MetaDatumItem](#) with the **largest value** of attribute [length](#) is **located in the first position** and the [MetaDatumItem](#) with the **smallest value** of attribute [length](#) is **located in the last position**.

This rule shall be imposed **at the time when the contract phase generation is executed.** ]()

**[TPS\_SWCT\_01803] MetaDatumItems define the same value of attribute [length](#)** [For the case that [MetaDatumItems](#) define the same value of attribute [length](#) the ordering of these [MetaDatumItems](#) among each other is not significant.] ([RS\\_SWCT\\_02030](#))

**[TPS\_SWCT\_01804] Standardized values of attribute [MetaDatumItem.metaDatumItemType.value](#)** [The following values of attribute [MetaDatumItem.metaDatumItemType.value](#) are standardized by AUTOSAR:

- PRIORITY\_8
- SOURCE\_ADDRESS\_16
- TARGET\_ADDRESS\_16
- ADDRESS\_EXTENSION\_8
- SOCKET\_CONNECTION\_ID\_16
- LIN\_NAD\_8
- CAN\_ID\_32
- ETHERNET\_MAC\_32

]()

**[constr\_1866] Existence of [MetaDatumItem.length](#)** [For each [MetaDatumItem](#), attribute [length](#) shall exist **at the time when the contract phase generation is executed.** ]()

**[constr\_1867] Existence of [MetaDatumItem.metaDatumItemType](#)** [For each [MetaDatumItem](#), attribute [metaDatumItemType](#) shall exist **at the time when the contract phase generation is executed.** ]()

**[constr\_1868] Existence of `MetaDataItemSet.dataElement`** [For each `MetaDataItemSet` that aggregates at least one `metaDataItem`, at least one reference to a `dataElement` shall exist **at the time when the contract phase generation is executed.**]()

### 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 **at the time when the RTE is generated.**]()

#### 4.2.3.1 Client Server Interface

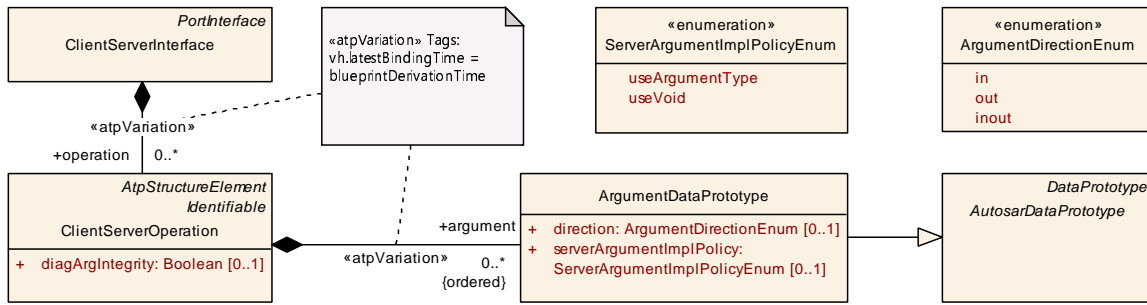
A `ClientServerInterface`, to some extent, is a counterpart to the `SenderReceiverInterface`<sup>2</sup>.

Instead of defining pieces of information to be transferred among software-components, a `ClientServerInterface` defines a collection of `ClientServerOperations`.

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

**Table 4.8: ClientServerInterface**

<sup>2</sup>However, different connection patterns apply, see [\[constr\\_1037\]](#)



**Figure 4.2: ClientServerOperations of a ClientServerInterface**

**[TPS\_SWCT\_01118] ClientServerInterface** [A **ClientServerInterface** is composed of **ClientServerOperations**, i.e. a **ClientServerOperation** cannot be reused in the context of a different **ClientServerInterface**.] ([RS\\_SWCT\\_02030](#))

**[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\\_02030](#), [RS\\_SWCT\\_03141](#))

**[TPS\_SWCT\_01844] Optional method arguments** [AUTOSAR does not support the existence of optional **arguments** within a **ClientServerOperation**.] ( )

The reason for the existence of the restriction in [\[TPS\\_SWCT\\_01844\]](#) on the *AUTOSAR classic platform* is that the RTE does not have an API to handle optional method arguments.

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, MultilanguageReferrable, Referrable			
Attribute	Type	Mult.	Kind	Note
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime







Class	ClientServerOperation			
diagArgIntegrity	Boolean	0..1	attr	This attribute shall only be used in the implementation of diagnostic routines to support the case where input and output arguments are allocated in a shared buffer and might unintentionally overwrite input arguments by tentative write operations to output arguments.  This situation can happen during sliced execution or while output parameters are arrays (call by reference). The value true means that the ClientServerOperation is aware of the usage of a shared buffer and takes precautions to avoid unintentional overwrite of input arguments.  If the attribute does not exist or is set to false the Client ServerOperation does not have to consider the usage of a shared buffer.
possibleError	<a href="#">ApplicationError</a>	*	ref	Possible errors that may be raised by the referring operation.

**Table 4.9: 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	<a href="#">ARObject</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">AutosarDataPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> , <a href="#">Referrable</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
direction	<a href="#">ArgumentDirection</a> <a href="#">Enum</a>	0..1	attr	This attribute specifies the direction of the argument prototype.
serverArgument ImplPolicy	<a href="#">ServerArgumentImpl</a> <a href="#">PolicyEnum</a>	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.

**Table 4.10: 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).] ([RS\\_SWCT\\_02030](#))

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](#).] ([RS\\_SWCT\\_02030](#))

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](#)).

**[constr\_1869] Existence of attribute `ArgumentDataPrototype.direction`** [For each `ArgumentDataPrototype`, attribute `direction` shall be defined **at the time when the contract phase generation is executed.**](*)*

**[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`.](*RS\_SWCT\_02030*)

**[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`.](*RS\_SWCT\_02030*)

**[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`.](*RS\_SWCT\_02030*)

<i>Enumeration</i>	<b>ArgumentDirectionEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
<b>Note</b>	Use cases: <ul style="list-style-type: none"> <li>Arguments in <code>ClientServerOperation</code> can have different directions that need to be formally indicated because they have an impact on how the function signature looks like eventually.</li> <li>Arguments in <code>BswModuleEntry</code> already determine a function signature, but the direction is used to specify the semantics, especially of pointer arguments.</li> </ul>
<b>Literal</b>	<b>Description</b>
in	The argument value is passed to the callee. <b>Tags:</b> atp.EnumerationLiteralIndex=0
inout	The argument value is passed to the callee but also passed back from the callee to the caller. <b>Tags:</b> atp.EnumerationLiteralIndex=1
out	The argument value is passed from the callee to the caller. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 4.11: 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)<sup>3</sup>.

<sup>3</sup>In 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`

**[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* argument<sup>4</sup>.] (*RS\_SWCT\_02030*)

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`.

<b>Enumeration</b>	<code>ServerArgumentImplPolicyEnum</code>
<b>Package</b>	<code>M2::AUTOSARTemplates::SWComponentTemplate::PortInterface</code>
<b>Note</b>	This defines how the argument type of the servers <code>RunnableEntity</code> is implemented.
<b>Literal</b>	<b>Description</b>
<code>useArgumentType</code>	The argument type of the <code>RunnableEntity</code> is derived from the <code>AutosarDataType</code> of the <code>ArgumentPrototype</code> . <b>Tags:</b> <code>atp.EnumerationLiteralIndex=0</code>
<code>useVoid</code>	The argument type of the <code>RunnableEntity</code> is void. <b>Tags:</b> <code>atp.EnumerationLiteralIndex=2</code>

**Table 4.12: `ServerArgumentImplPolicyEnum`**

**[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]).

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

There is one use case for executing multiple `ClientServerOperations` using the exact same `RunnableEntity`. This use case is sketched in Figure 4.3.

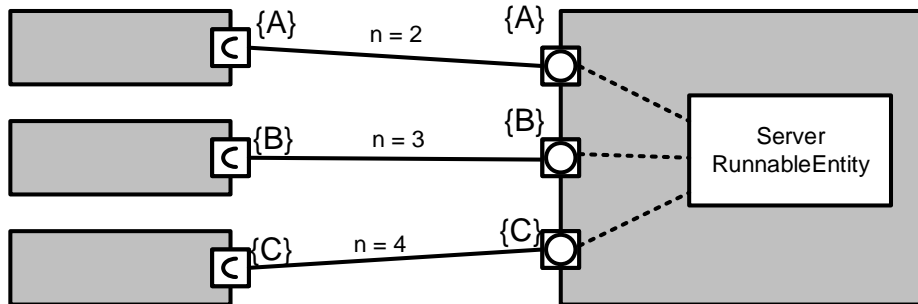
<sup>4</sup>Giving the `ArgumentDataPrototypes` of a `ClientServerOperation` both an ordering and a unique identifier might seem redundant.

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.

In such a case, it may happen that the `ClientServerOperations` own `ArgumentDataPrototypes` that are typed by an array data type.

It may also happen that the individual `ArgumentDataPrototypes` that are typed by array data types refer to a compatible element data type, but define a different number of elements.



**Figure 4.3: Use case for one server `RunnableEntity` processes calls from different `PortPrototypes`**

Please note that the server `RunnableEntity` needs information about the currently used array length respectively structure size by usage of additional arguments passed by the Client. This aspect is exemplified by Figure 4.3.

It is only natural that in such a case a `Variable-Size Array Data Type` would be used because it comes with a built-in capability to indicate the number of elements currently stored in the array without the need to add further arguments to the signature of the `RunnableEntity`.

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”).

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

**Table 4.13: Supported connections by `AssemblySwConnector` for `PortPrototypes` typed by a `ClientServerInterface`, `ModeSwitchInterface`, or `TriggerInterface`**

**[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 4.13 are supported by AUTOSAR **at any time in the workflow.**]()

**[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 4.14 are supported by AUTOSAR **at any time in the workflow.**]()

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

**Table 4.14: Supported connections by DelegationSwConnector for PortPrototypes typed by a ClientServerInterface, ModeSwitchInterface, or TriggerInterface**

Please note that a restriction concerning the usage of attribute ClientServerOperation.diagArgIntegrity is described in section 13.8.4.4.

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

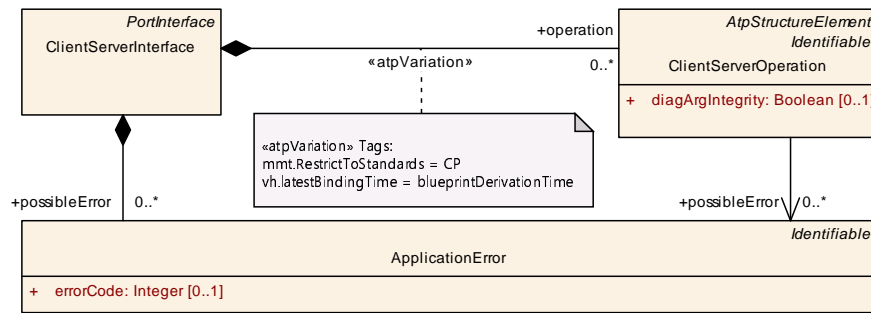


Figure 4.4: Application error meta-model

[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`.

[constr\_1102] **ApplicationError in the scope of one SwComponentType** [If a `SwComponentType` has `PortPrototypes` typed by different `ClientServerInterfaces` with equal `shortName` and `ApplicationErrors` defined then the following condition applies: `ApplicationErrors` with the same `shortName` shall have **identical values** of `errorCodes`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Rationale for the existence of [constr\_1102]: the RTE generator creates symbols for the error codes in which the `shortName` of the `ClientServerInterface` and the `shortName` of the `ApplicationError` occur.

[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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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.

<b>Class</b>	<b>ApplicationError</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
errorCode	Integer	0..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 4.15: 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 [PortInterface](#) that also owns the [ClientServerOperation](#) **at any time in the workflow.**]()

Please note that the meta-class [ApplicationError](#) is also used on the *AUTOSAR adaptive platform* (see [14]) and therefore **[constr\_1038]** cannot be more specific about the nature of the enclosing [PortInterface](#).

**[constr\_1870] Existence of attribute [ApplicationError.errorCode](#)** [For each [ApplicationError](#), attribute [errorCode](#) shall be defined **at the time when the contract phase generation is executed.**]()

#### 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 [RunnableEntities](#) in the connected trigger sinks. Typically, (but not necessarily) these [RunnableEntities](#) are executed in a sequential order.]([RS\\_SWCT\\_02030](#))

**[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.]()

<b>Class</b>	<b>TriggerInterface</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
<b>Note</b>	A trigger interface declares a number of triggers that can be sent by an trigger source. <b>Tags:</b> atp.recommendedPackage=PortInterfaces





<b>Class</b>	<b>TriggerInterface</b>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
trigger	<a href="#">Trigger</a>	*	aggr	The Trigger of this trigger interface.

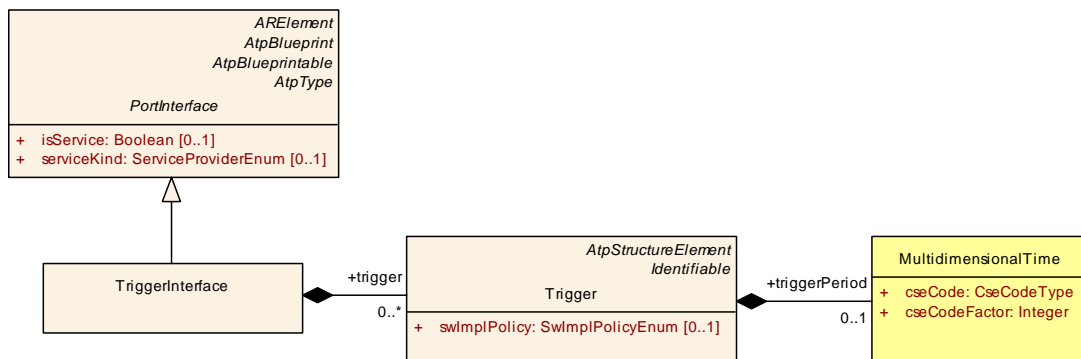
**Table 4.16: TriggerInterface**

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

**Table 4.17: Trigger**

<b>Class</b>	<b>MultidimensionalTime</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::MultidimensionalTime			
<b>Note</b>	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 0 and the cseCodeFactor is 50 it represents 50 microseconds.			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
cseCode	<a href="#">CseCodeType</a>	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 4.18: 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.] (*RS\_SWCT\_02030*)

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 at the time when the RTE is generated.**] ()

[*constr\_1104*] boils down to the requirement that trigger communication shall not be implemented in an n:1 (where n > 1) scenario.

To be clear, the n:1 (where 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 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).

**at the time when the contract phase generation is executed.**] ()

Please note that the value of `Trigger.swImplPolicy` is not the final word on the implementation of a queue for the specific `Trigger`. The integrator still has the power to overrule the application software developer's verdict if applicable.

For more information regarding the ability to connect different kinds of `PortPrototypes` typed by a `TriggerInterface` to each other 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 it uses a special type of `PortInterface`: the collections of `ModeDeclarations` that are required or provided by a `SwComponentType` are defined by means of `ModeSwitchInterfaces` used to type the `PortPrototypes` owned by the `SwComponentType`.] ([RS\\_SWCT\\_02030](#), [RS\\_SWCT\\_03203](#))

This aspect is depicted in Figure 4.6.

**[constr\_2049] Different `ModeDeclarationGroups` shall have different `shortNames`.** [A software component is not allowed to type multiple `PortPrototypes` with `ModeSwitchInterfaces` where the contained `ModeDeclarationGroupPrototypes` are referencing `ModeDeclarationGroups` with identical `shortNames` but different `ModeDeclarations`.

**at the time when the contract phase generation is executed**] ()

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 the `ModeDeclarations` “W”, “X”, “Y”, “Z”

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

<b>Class</b>	<b>ModeSwitchInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	A mode switch interface declares a <code>ModeDeclarationGroupPrototype</code> to be sent and received. <b>Tags:</b> atp.recommendedPackage=PortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





<b>Class</b>	<b>ModeSwitchInterface</b>			
modeGroup	<a href="#">ModeDeclarationGroupPrototype</a>	0..1	aggr	The ModeDeclarationGroupPrototype of this mode interface.

**Table 4.19: ModeSwitchInterface**

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

**Table 4.20: 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](#).

This rule shall be imposed **at any time in the workflow.**]( )

**[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 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 5.43.

**[TPS\_SWCT\_01200] ModeDeclarationGroupPrototype per ModeSwitchInterface** [The multiplicity of the aggregation of ModeDeclarationGroupPrototype to ModeSwitchInterface is pragmatically limited to 1.](RS\_SWCT\_03203)

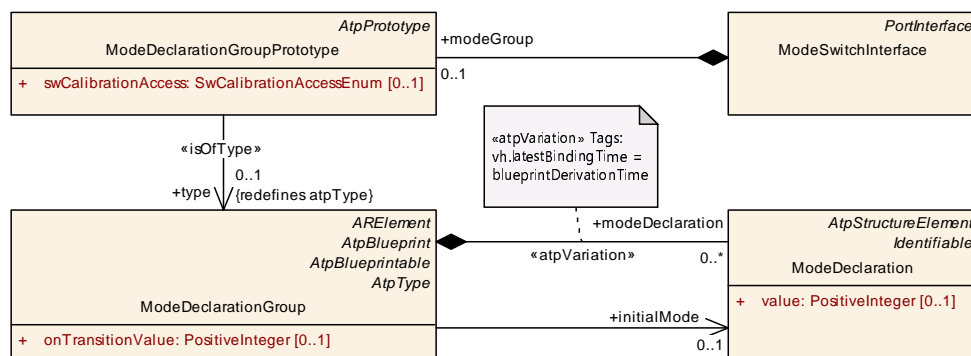
Admittedly, there would be no technical restriction to support a 0..\* multiplicity but on the other hand it does not seem as if any reasonable use case for such a scenario exists.

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.

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



**Figure 4.6: Mode Switch Interface**

**[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](#) at the time when the contract phase generation is executed.] ()

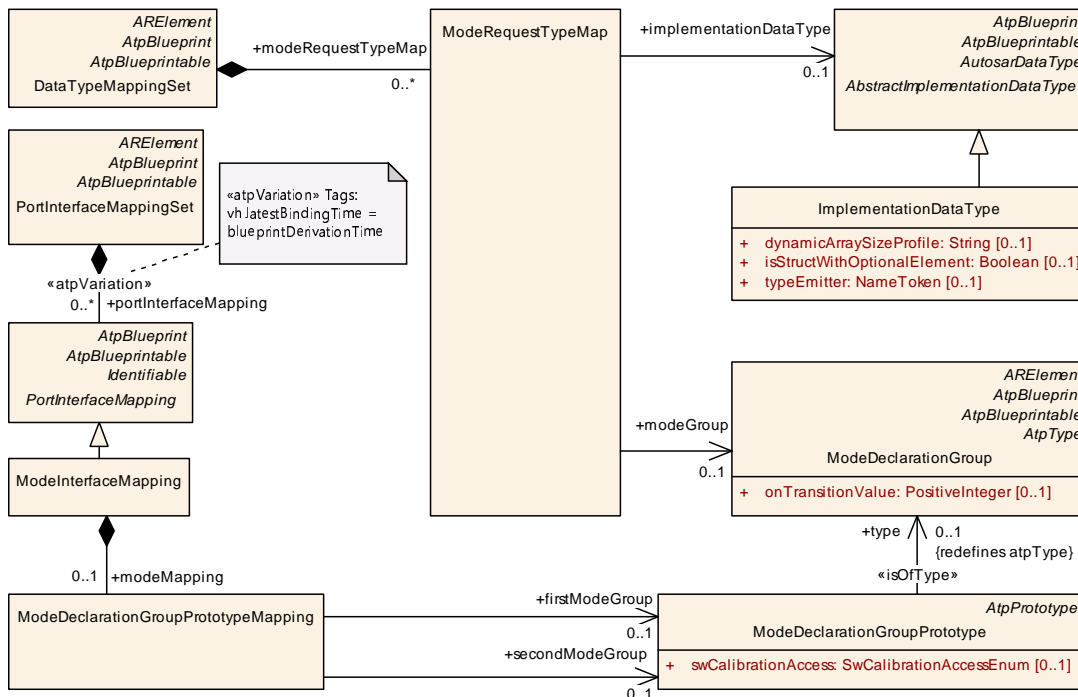


Figure 4.7: Mapping of modes to data types

<b>Class</b>	<b>ModeRequestTypeMap</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	Specifies a mapping between a ModeDeclarationGroup and an ImplementationDataType. This ImplementationDataType shall be used to implement the ModeDeclarationGroup.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





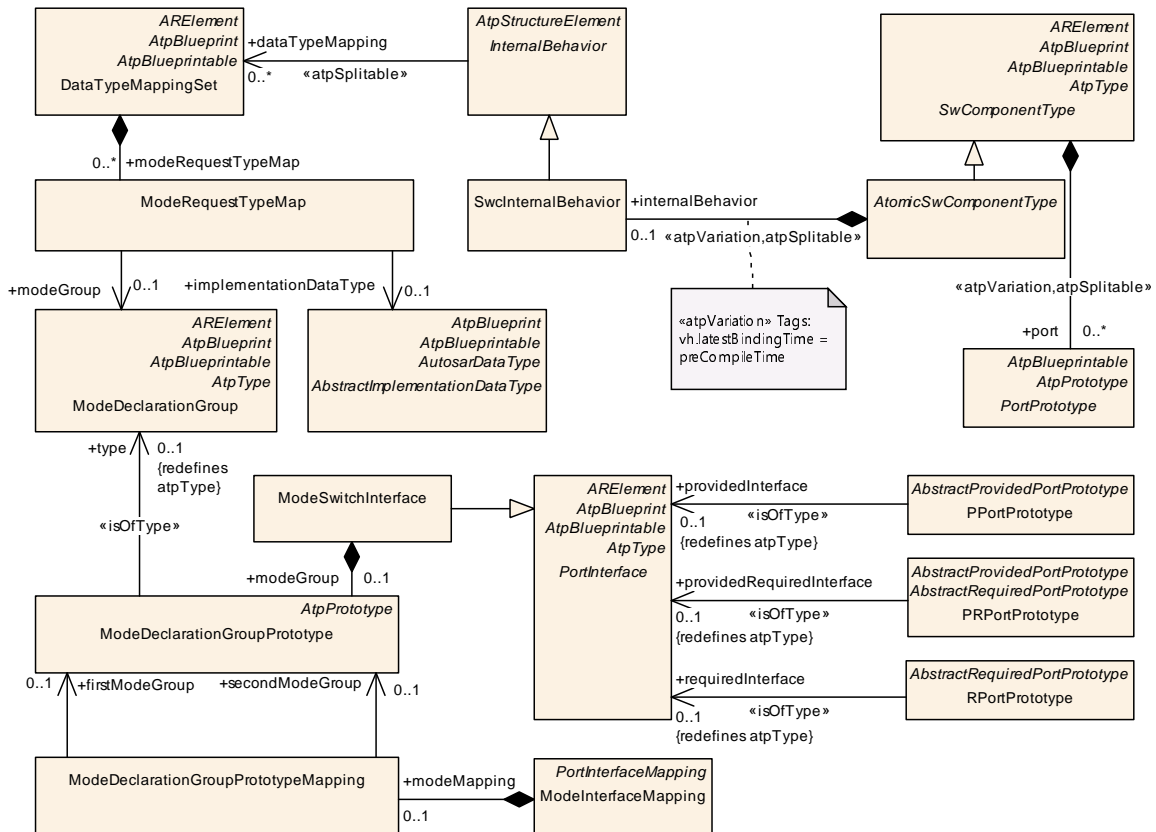
Class	ModeRequestTypeMap			
implementation DataType	<a href="#">AbstractImplementation DataType</a>	0..1	ref	This is the corresponding AbstractImplementationDataType. It shall be modeled along the idea of an "unsigned integer-like" data type.
modeGroup	<a href="#">ModeDeclarationGroup</a>	0..1	ref	This is the corresponding ModeDeclarationGroup.

**Table 4.21: 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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()



**Figure 4.8: Big picture of mode declaration mapping**

**[constr\_1871] Existence of attribute ModeRequestTypeMap.implementationDataType** [For each ModeRequestTypeMap, attribute implementationDataType

shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1872] Existence of attribute `ModeRequestTypeMap.modeGroup`** [For each `ModeRequestTypeMap`, attribute `modeGroup` shall exist **at the time when the contract phase generation is executed.**]()

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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 **at any time in the workflow.** ]()

### 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 other but don't fulfill formal requirements the concept of “port interface mapping” has been introduced.

**[TPS\_SWCT\_01158] Cases for PortInterfaceMapping** [In general, the existence of a `PortInterfaceMapping` is suitable in the following cases:

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](#))

More information about the AUTOSAR data transformer approach can be found in section 4.3.3.

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

**[TPS\_SWCT\_01159] Mapping is described separately from the SwConnector as reusable ARElement** [The mapping is described separately from the `SwConnector` 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.

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

**Table 4.22: PortInterfaceMappingSet**

<b>Class</b>	<b>PortInterfaceMapping</b> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
<b>Note</b>	Specifies one <a href="#">PortInterfaceMapping</a> to support the connection of Ports typed by two different <a href="#">PortInterfaces</a> with <a href="#">PortInterface</a> elements having unequal names and/or unequal semantic (resolution or range).
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>





<b>Class</b>	<i>PortInterfaceMapping</i> (abstract)			
<b>Subclasses</b>	ClientServerInterfaceMapping, ModelInterfaceMapping, TriggerInterfaceMapping, VariableAndParameterInterfaceMapping			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 4.23: 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.

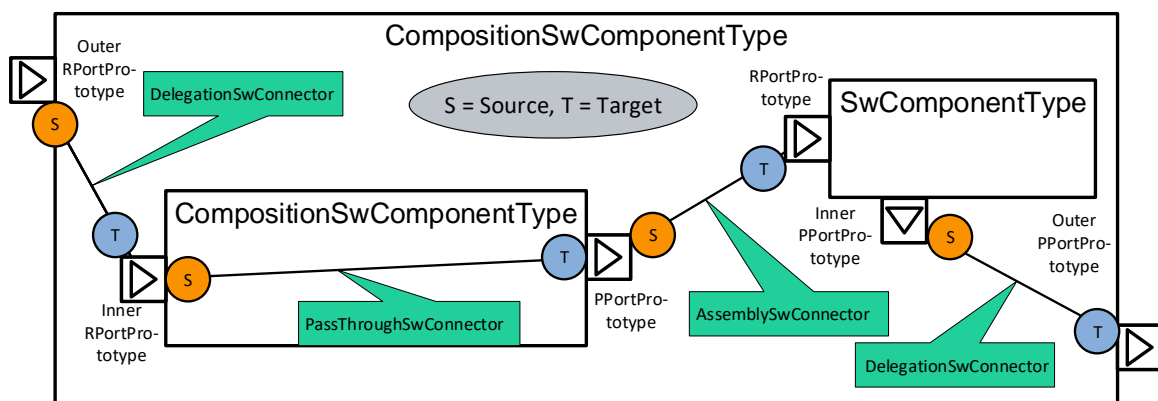
This definition is provided with `PortInterfaceMappings`.

Please note that the `PortInterfaceMapping` is a very powerful tool for expert users to accommodate for various differences in the `PortInterfaces` of two `PortPrototypes` that shall be connected.

In general, it is possible to define a `PortInterfaceMapping` for all sub-classes of `SwConnectors`. To make this possible, it is necessary to avoid talking about “provided” or “required” ends because this could lead to confusion.

Specifically, it is possible that inside a `CompositionSwComponentType` a `PassThroughSwConnector` is connected to a `PPortPrototype`.

From the perspective of the `PassThroughSwConnector`, however, the `PPortPrototype` might represent the “required” end of the communication relation inside the `CompositionSwComponentType`.



**Figure 4.9: “Source” and “Target” end of a SwConnector in a sender/receiver-style scenario**

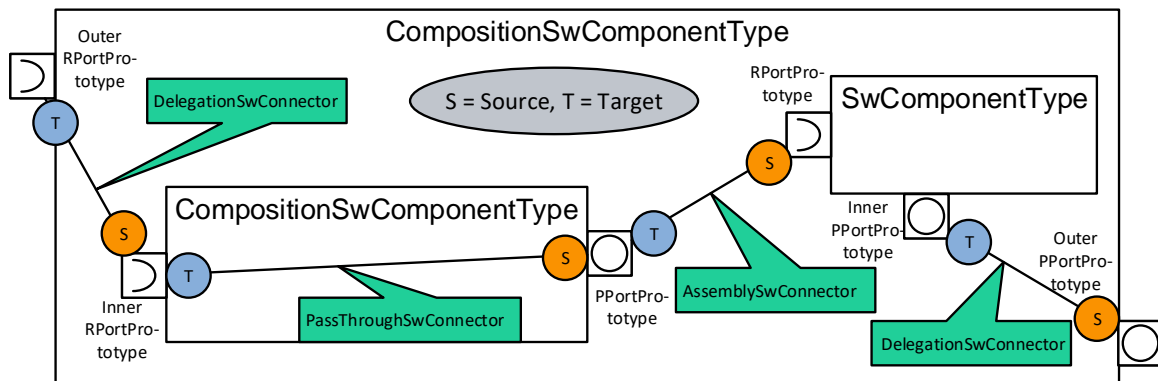
To avoid further confusion, the rules for the specification of the `PortInterfaceMapping` are specified **from the perspective of the `SwConnector` rather than the `PortPrototype`**.

For each `SwConnector`, a “source” and “target” end is postulated, reflecting the direction of communication implemented by the `SwConnector`. This understanding is depicted in Figure 4.9.

Please note that the “sender/receiver-style communication” (see section 4.2.2) is conceptually also applicable for other kinds of communication:

- Mode switch communication<sup>5</sup>, see section 4.2.5
- External Trigger communication, see section 4.2.4
- NvData communication, see section 11.5.2
- Parameter “communication”, see 4.2.6

Note further that the association of the “source” and “target” end of an `SwConnector` that represents a client/server communication (see section 4.2.3) is aligned along the “primary” interaction, i.e. the call of a `ClientServerOperation`, see Figure 4.10.



**Figure 4.10: “Source” and “Target” end of a `SwConnector` in a client/server-style scenario**

Sender/Receiver	Source	Target
<code>DelegationSwConnector</code>	Outer required <code>PortPrototype</code>	Inner required <code>PortPrototype</code>
<code>DelegationSwConnector</code>	Inner provided <code>PortPrototype</code>	Outer provided <code>PortPrototype</code>
<code>AssemblySwConnector</code>	Provided <code>PortPrototype</code>	Required <code>PortPrototype</code>
<code>PassThroughSwConnector</code>	Required <code>PortPrototype</code>	Provided <code>PortPrototype</code>

**Table 4.24: Definition of “source” and “target” of an `SwConnector` for a sender/receiver-style interaction**

<sup>5</sup>A mode request is implemented by a sender/receiver communication anyway and does therefore not represent a dedicated communication pattern.

The conclusion of Figure 4.9 regarding “source” and “target” end of an `SwConnector` for a sender/receiver-style interaction is summarized in Table 4.24.

The conclusion of Figure 4.10 regarding “source” and “target” end of an `SwConnector` for a client/server-style interaction is summarized in Table 4.25.

Client/Server	Source	Target
<code>DelegationSwConnector</code>	Inner required <code>PortPrototype</code>	Outer required <code>PortPrototype</code>
<code>DelegationSwConnector</code>	Outer provided <code>PortPrototype</code>	Inner provided <code>PortPrototype</code>
<code>AssemblySwConnector</code>	Required <code>PortPrototype</code>	Provided <code>PortPrototype</code>
<code>PassThroughSwConnector</code>	Provided <code>PortPrototype</code>	Required <code>PortPrototype</code>

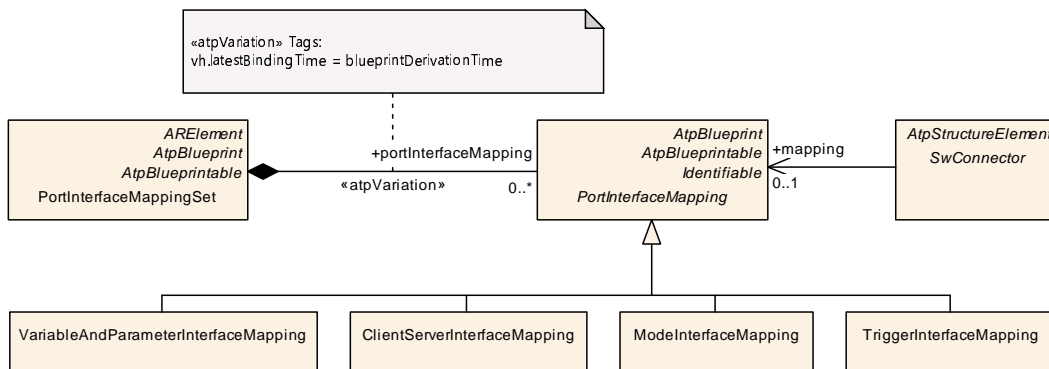
**Table 4.25: Definition of “source” and “target” of an `SwConnector` for a client/server-style interaction**

There are some documented restrictions for the application of `PortInterfaceMappings`. But in general, the `PortInterfaceMapping` could potentially be used in a harmful way by specifying mapping approaches that have no chance to be implementable in the context of an ECU, resulting in undefined semantics in the RTE, i.e. the RTE Generator wouldn't be able to properly process such a model.

**[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`. This rule shall be imposed **at the time when the RTE is generated.**] ()



**Figure 4.11: Relevant meta-classes for `PortInterface` element mapping**

**[TPS\_SWCT\_01100] Precedence of `PortInterfaceMapping`** [The mapping via `PortInterfaceMapping` has a higher precedence than the mapping by equal `shortNames` as defined in compatibility rules.]

If a connector references an associated `PortInterfaceMapping`, this mapping shall be strictly binding with respect to the number of mapped data elements.]([RS\\_SWCT\\_03155](#), [RS\\_SWCT\\_03210](#))

Please note that the compatibility rules are described in chapter 6.

**[TPS\_SWCT\_01101] Unmapped elements of `PortInterfaces`** [Unmapped `PortInterface` elements will not be connected by the referencing `SwConnector`.]([RS\\_SWCT\\_03155](#), [RS\\_SWCT\\_03210](#))

**[constr\_1583] `PortInterfaceMapping` for `DataPrototype` typed by `Compound Primitive Data Type`** [There is one very limited use case to apply `PortInterfaceMapping` for a `DataPrototype` typed by a `Compound Primitive Data Type`: adjustment of the `shortName` of the `DataPrototype`. Everything else is **not supported**. This rule shall be imposed **at any time in the workflow**.]()

#### 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` **at any time in the workflow**.]()

Please note that the relation of the “source” and “target” end of a `SwConnector` to the roles `firstDataPrototype` and `secondDataPrototype` is not defined.

**[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](#))

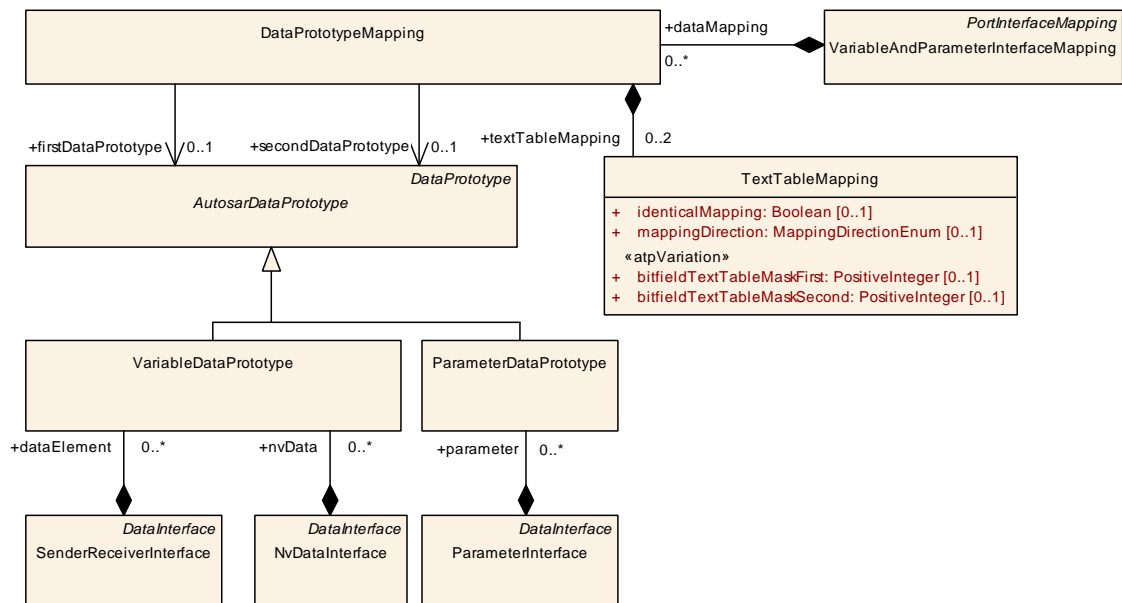
For more explanation of [TPS\_SWCT\_01104], please refer to [constr\_1071].

**[constr\_1039] Relevance of `swImplPolicy`** [It is not possible to define a mapping between an element where the `swImplPolicy` is set to `queued` and another element

where the `swImplPolicy` is set differently. This rule shall be imposed **at any time in the workflow.** ]()

This is required to fulfill the compatibility rules defined in table 6.2.

**[constr\_1635] Relevance of attribute `isOptional`** [If a `SubElementMapping` is defined for the elements of a structured data type then the attribute `isOptional`<sup>6</sup> shall either not exist for the `firstElement` and `secondElement` or it shall have the identical value for the `firstElement` and `secondElement`. This rule shall be imposed **at the time when the RTE is generated.** ]()



**Figure 4.12: Mapping of Sender Receiver Interface, Parameter Interface and Non Volatile Data Interface elements**

**[constr\_1040] Conversion of `SenderReceiverInterfaces`** [The conversion of elements of `SenderReceiverInterfaces` is possible if one of the following conditions applies **at the time when the RTE is generated:**

- The `AutosarDataTypes` of the referred `DataPrototypes` are compatible.
- A conversion of the data is available.
- A `DataPrototypeMapping.firstToSecondDataTransformation` is defined.

]()

The compatibility of `AutosarDataTypes` is described in section 6.2. A description of the conversion of data can be found in section 4.3.2.

<sup>6</sup>this is valid for both `ApplicationRecordElement` and `ImplementationDataTypeElement`

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

**Table 4.26: VariableAndParameterInterfaceMapping**

<b>Class</b>	<b>DataPrototypeMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	<p>Defines the mapping of two particular VariableDataPrototypes, ParameterDataPrototypes or Argument DataPrototypes with non-equal shortNames, non-equal structure (specific condition is described by [constr_1187]), and/or non-equal semantic (resolution or range) in context of two different Sender ReceiverInterface, NvDataInterface or ParameterInterface or Operations.</p> <p>If the semantic is unequal, the 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.</p> <p>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.</p>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
firstData Prototype	AutosarDataPrototype	0..1	ref	First to be mapped DataPrototype in context of a Sender ReceiverInterface, NvDataInterface, ParameterInterface or Operation.
firstToSecond Data Transformation	DataTransformation	0..1	ref	<p>This reference defines the need to execute the Data Transformation &lt;Mip&gt;_&lt;transformerId&gt; functions of the transformation chain when communicating from the Data PrototypeMapping.firstDataPrototype to the Data PrototypeMapping.secondDataPrototype.</p> <p>This reference also specifies the reverse Data Transformation &lt;Mip&gt;_Inv_&lt;transformerId&gt; functions of the transformation chain (i.e. from the DataPrototype Mapping.secondDataPrototype to the DataPrototype Mapping.firstDataPrototype) if the referenced Data Transformation is symmetric, i.e. attribute Data Transformation.dataTransformationKind is set to symmetric.</p>
secondData Prototype	AutosarDataPrototype	0..1	ref	Second to be mapped DataPrototype in context of a SenderReceiverInterface, NvDataInterface, Parameter Interface or Operation.
secondToFirst Data Transformation	DataTransformation	0..1	ref	This defines the need to execute the reverse Data Transformation <Mip>_Inv_<transformerId> functions of the transformation chain when communicating from the DataPrototypeMapping.secondDataPrototype to the Data PrototypeMapping.firstDataPrototype.
subElement Mapping	SubElementMapping	*	aggr	This represents the owned SubelementMapping.





Class	DataPrototypeMapping			
textTable Mapping	TextTableMapping	0..2	aggr	Applied TextTableMapping(s)

**Table 4.27: DataPrototypeMapping**

**[constr\_1873] Existence of DataPrototypeMapping.firstDataPrototype**  
[For each DataPrototypeMapping, the reference in the role firstDataPrototype shall exist **at the time when the RTE is generated.**]()

**[constr\_1874] Existence of DataPrototypeMapping.secondDataPrototype**  
[For each DataPrototypeMapping, the reference in the role secondDataPrototype shall exist **at the time when the RTE is generated.**]()

#### 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 two ClientServerInterfaces.](RS\_SWCT\_03155, RS\_SWCT\_03210)

**[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.

This rule shall be imposed **at any time in the workflow.**]()

**[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.

This rule shall be imposed **at any time in the workflow.**]()

**[constr\_1041] Conversion of ClientServerInterfaces** [Either the AutosarDataTypes of the referred ArgumentDataPrototypes are compatible or a conversion of the data is available **at the time when the RTE is generated.**]()

The compatibility of AutosarDataTypes is described in section 6.2. A description of the conversion of data can be found in section 4.3.2.



**[constr\_1240] Consistency of `ArgumentDataPrototypes` within the context of a `ClientServerOperationMapping`** [Unless a `ClientServerOperationMapping.firstToSecondDataTransformation` exists, 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 **at any time in the workflow**, 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`, the value of the argument `ArgumentDataPrototype.direction` of two mapped `ArgumentDataPrototype` shall be identical **at any time in the workflow**.]()

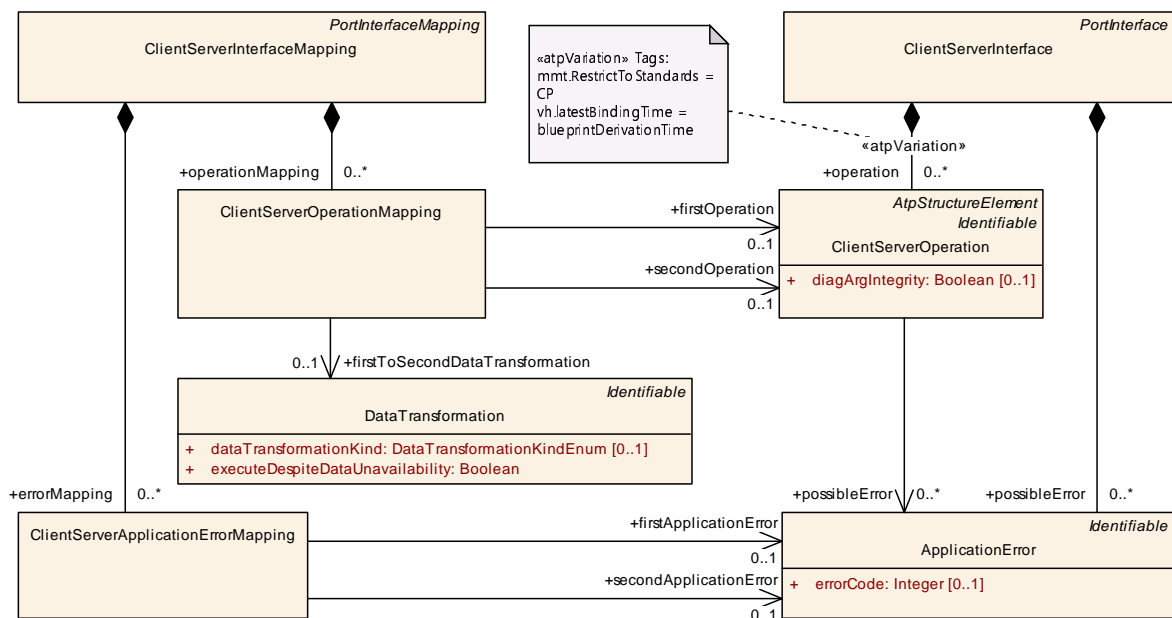


Figure 4.13: Mapping of `ClientServerInterface` elements and mapping of arguments

**[constr\_1269] Number of `arguments` shall be preserved in a `ClientServerOperationMapping`** [Within the context of a `ClientServerOperationMapping`, the number of `arguments` of `firstOperation` and `secondOperation` shall be identical **at any time in the workflow**.]()

[constr\_1270] **ArgumentDataPrototype** shall be mapped only once in a **ClientServerOperationMapping** [Within the context of a **ClientServerOperationMapping**, each **argument** shall only be referenced **once** in the role **firstDataPrototype** or **secondDataPrototype** **at any time in the workflow.**]

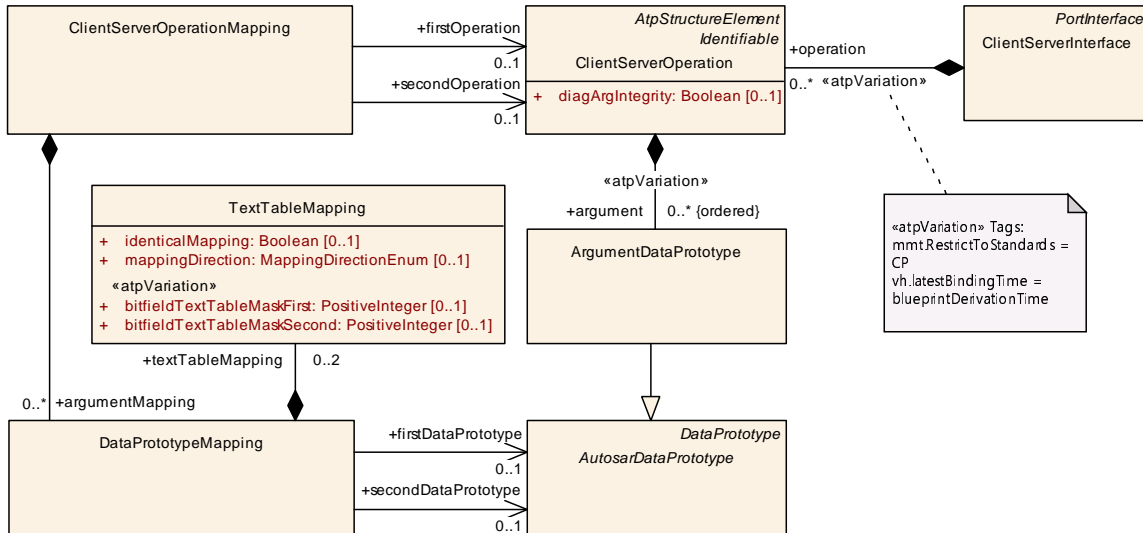


Figure 4.14: Mapping of **ArgumentDataPrototypes**

[constr\_1469] **Applicability of constraints depending on the existence of a data transformation** [[constr\_1269], [constr\_1270], [constr\_1268], and [constr\_1240] shall – **at any time in the workflow** – **not** apply under the following conditions:

- A reference from the respective **ClientServerOperationMapping** to a **DataTransformation** in the role **firstToSecondDataTransformation** exists.
- The value of the attribute **dataTransformationKind** of the referenced **DataTransformation** is set to **DataTransformationKindEnum.asymmetricFromByteArray** or **DataTransformationKindEnum.asymmetricToByteArray**.

]()

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

Table 4.28: ClientServerInterfaceMapping

<b>Class</b>	<b>ClientServerOperationMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Defines the mapping of two particular ClientServerOperations in context of two different ClientServer Interfaces.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
argument Mapping	DataPrototypeMapping	*	aggr	Defines the mapping of two particular ArgumentData Prototypes with unequal names or unequal semantic (resolution or range) in context of Operations.
firstOperation	ClientServerOperation	0..1	ref	First to-be-mapped ClientServerOperation of a Client ServerInterface.
firstToSecond Data Transformation	DataTransformation	0..1	ref	This reference indicates that a DataTransformation is intended in the context of the ClientServerOperation Mapping.
second Operation	ClientServerOperation	0..1	ref	Second to-be-mapped ClientServerOperation of a Client ServerInterface.

**Table 4.29: ClientServerOperationMapping**

<b>Class</b>	<b>ClientServerApplicationErrorMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	This meta-class represents the ability to map ApplicationErrors onto each other.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
firstApplication Error	ApplicationError	0..1	ref	This represents the first ApplicationError in the context of the ClientServerApplicationErrorMapping.
second ApplicationError	ApplicationError	0..1	ref	This represents the second ApplicationError in the context of the ClientServerApplicationErrorMapping.

**Table 4.30: ClientServerApplicationErrorMapping**

**[constr\_1875] Existence of reference ClientServerOperationMapping.firstOperation** [For each ClientServerOperationMapping, the reference in the role firstOperation shall exist **at the time when the RTE is generated.**]()

**[constr\_1876] Existence of reference ClientServerOperationMapping.secondOperation** [For each ClientServerOperationMapping, the reference in the role secondOperation shall exist **at the time when the RTE is generated.**]()

### 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`.] ([RS\\_SWCT\\_03210](#))

The compatibility of `ModeDeclarationGroups` is described in chapter 6.7.

<b>Class</b>	<b>ModeInterfaceMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Defines the mapping of <code>ModeDeclarationGroupPrototypes</code> in context of two different <code>ModeInterfaces</code> .			
<b>Base</b>	<i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PortInterfaceMapping</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
modeMapping	<a href="#">ModeDeclarationGroupPrototypeMapping</a>	0..1	aggr	Mapping of two <code>ModeDeclarationGroupPrototypes</code> in two different <code>ModeInterfaces</code>

**Table 4.31: ModeInterfaceMapping**

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

**Table 4.32: ModeDeclarationGroupPrototypeMapping**

[constr\_1877] **Existence of reference `ModeDeclarationGroupPrototypeMapping.firstModeGroup`** [For each `ModeDeclarationGroupPrototypeMapping`, the reference in the role `firstModeGroup` shall exist **at the time when the RTE is generated.**]()

[constr\_1878] **Existence of reference `ModeDeclarationGroupPrototypeMapping.secondModeGroup`** [For each `ModeDeclarationGroupPrototypeMapping`, the reference in the role `secondModeGroup` shall exist **at the time when the RTE is generated.**]()

[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](#))

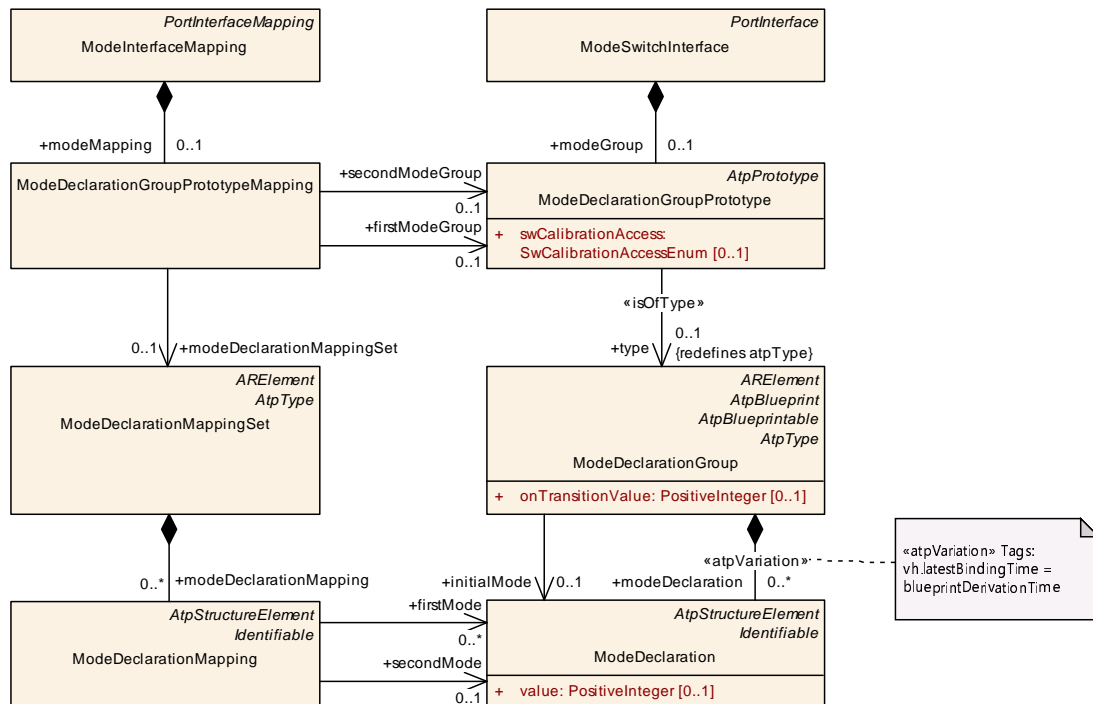


Figure 4.15: Mapping of **ModeSwitchInterface** elements

[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**

at any time in the workflow.]()

[constr\_1247] Consistency of **ModeDeclarationMappingSet** with respect to the referenced **firstModeGroup** and **secondModeGroup** [If a **ModeDeclarationGroupPrototypeMapping.modeDeclarationMappingSet** exists, then

- 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**.

This rule shall be imposed at any time in the workflow.]()

[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] ModeDeclarationGroupPrototypeMapping.modeDeclarationMappingSet** defines the applicable set of **ModeDeclarationMappings** [The attribute **ModeDeclarationGroupPrototypeMapping.modeDeclarationMappingSet** defines the applicable set of **ModeDeclarationMappings** for the connection of **ModeDeclarationGroupPrototypes** typed by **ModeDeclarationGroups** with differently named **ModeDeclarations** and/or with a different number of **ModeDeclarations**.]()

<b>Class</b>	<b>ModeDeclarationMappingSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	This meta-class implements a container for ModeDeclarationGroupMappings <b>Tags:</b> atp.recommendedPackage=PortInterfaceMappingSets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
mode Declaration Mapping	<a href="#">ModeDeclarationMapping</a>	*	aggr	This represents the collection of ModeDeclaration Mappings owned by the enclosing ModeDeclaration MappingSet.

**Table 4.33: ModeDeclarationMappingSet**

<b>Class</b>	<b>ModeDeclarationMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	This meta-class implements a concrete mapping of two ModeDeclarations.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
firstMode	<a href="#">ModeDeclaration</a>	*	ref	This represents the first ModeDeclaration of the Mode DeclarationMapping. 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	<a href="#">ModeDeclaration</a>	0..1	ref	This represents the second ModeDeclaration of the Mode DeclarationMapping.

**Table 4.34: ModeDeclarationMapping**

**[constr\_1879] Existence of reference ModeDeclarationMapping.firstMode** [For each **ModeDeclarationMapping**, at least one reference **firstMode** shall exist **at the time when the RTE is generated**.]()

**[constr\_1880] Existence of reference ModeDeclarationMapping.secondMode** [For each **ModeDeclarationMapping**, the reference **secondMode** shall exist **at the time when the RTE is generated**.]()

**[TPS\_SWCT\_01464] ModeDeclaration of a mode user is mapped to exactly one ModeDeclaration of a mode manager** [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.] (**RS\_SWCT\_03115**)

**[TPS\_SWCT\_01465] ModeDeclaration of a mode user is mapped to several ModeDeclarations of a mode manager** [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 `ModeDeclarations` 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 **at any time in the workflow.**]()

**[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 **at the time when the RTE is generated.**]()

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*)

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

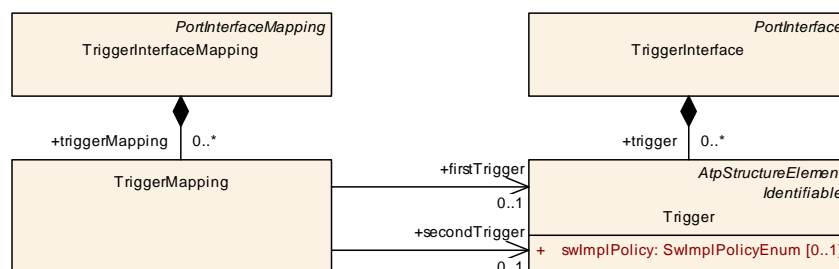
**Table 4.35: TriggerInterfaceMapping**

<b>Class</b>	<b>TriggerMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration			
<b>Note</b>	Defines the mapping of two particular unequally named Triggers in the given context.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
firstTrigger	Trigger	0..1	ref	A Trigger to be mapped.
secondTrigger	Trigger	0..1	ref	A Trigger to be mapped.

**Table 4.36: TriggerMapping**

**[constr\_1881] Existence of reference `TriggerMapping.firstTrigger`** [For each `TriggerMapping`, the reference `firstTrigger` shall exist **at the time when the RTE is generated.**]()

**[constr\_1882] Existence of reference `TriggerMapping.secondTrigger`** [For each `TriggerMapping`, the reference `secondTrigger` shall exist **at the time when the RTE is generated.**]()


**Figure 4.16: 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 other.

**[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 other.](RS\_SWCT\_03210, RS\_SWCT\_03135)

This ability (for which [constr\_10087] applies) can be used if e.g. dataElements on the “source” and “target” end side are typed by different ApplicationRecord-DataTypes.

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

**[constr\_10087] Restriction for the existence of a SubElementMapping** [The existence of a DataPrototypeMapping.subElementMapping is only supported if the PortPrototypes that are referenced by the respective SwConnector are typed by a DataInterface **at the time when the RTE is generated.**]( )

**[TPS\_SWCT\_01551] Mapping of elements on the “source” end to elements on the “target” end** [Unless the attribute swImplPolicy is set to queued, it is not required that all elements on the “source” end need to be mapped to elements on the “target” end 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, then it is not allowed to have unmapped elements of ApplicationCompositeDataTypes or ImplementationDataTypes of category STRUCTURE or ARRAY on the “target” end **at the time when the RTE is generated.**]( )

**[constr\_1280] Unmapped dataElement on the “target” end shall have an initialValue** [If elements of ApplicationCompositeDataTypes or ImplementationDataTypes of category STRUCTURE or ARRAY are not considered in a SubElementMapping **and** the NonqueuedReceiverComSpec is aggregated by an AbstractRequiredPortPrototype referenced by the “target” end, then the enclosing dataElement shall have an initialValue.

This rule shall be imposed **at the time when the RTE is generated.**]( )

**[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)

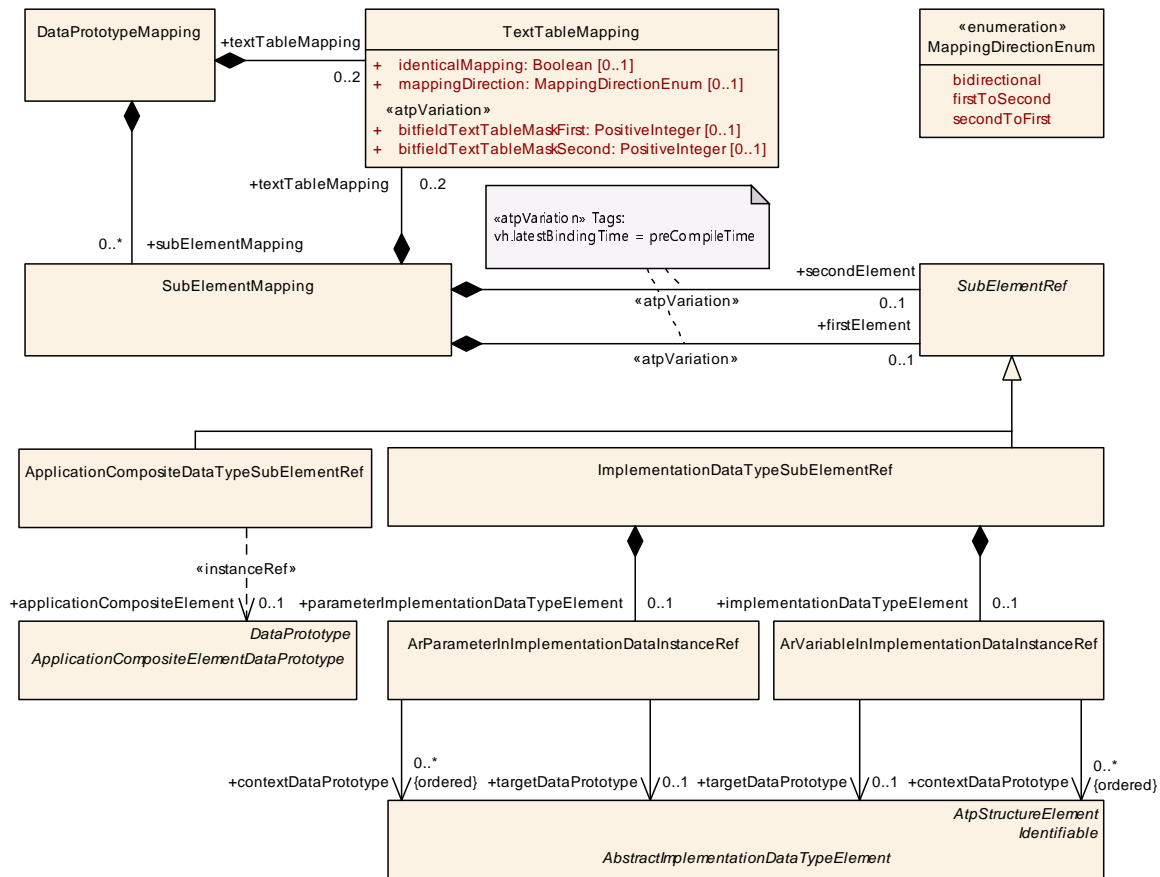


Figure 4.17: Mapping of elements of composite data types

**[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 “source” end to a primitive [DataPrototype](#) on the “target” end.]

For this purpose the multiplicity of the [dataElement/nvData](#) contained in the [PortPrototype](#) on the “source” end shall be set to 1 and the multiplicity of the [dataElement/nvData](#) contained in the [PortPrototype](#) on the “target” end shall be set to 0.]  
 ([RS\\_SWCT\\_03136](#))

In general, the multiplicity of the respective [dataElement/nvData](#) contained in the [PortPrototype](#) on the “source” end 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](#) at the time when the RTE is generated.]()

**[constr\_1300] Primitive [DataPrototype](#) on the “source” end shall not be mapped to element of a composite data type on the “target” end of the [SwConnector](#)** [The usage of [DataPrototypeMapping](#) or [SubElementMapping](#) does not support the following configuration:

- The `AutosarDataPrototype` contained in the `PortPrototype` on the “source” end of an `SwConnector` 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` contained in the `PortPrototype` on the “target” end.

This rule shall be imposed **at any time in the workflow.**]()

**[constr\_1611] Existence of `ImplementationDataTypeSubElementRef.implementationDataTypeElement` as opposed to `ImplementationDataTypeSubElementRef.parameterImplementationDataTypeElement`** [For any given `ImplementationDataTypeSubElementRef`, either the aggregation

- `ImplementationDataTypeSubElementRef.implementationDataTypeElement` or
- `ImplementationDataTypeSubElementRef.parameterImplementationDataTypeElement`

shall exist **at any time in the workflow.**]()

In other words, the `ImplementationDataTypeSubElementRef` shall **either** refer to the nested hierarchy inside a `VariableDataPrototype` **or** a `ParameterDataPrototype`.

<b>Class</b>	<b>SubElementMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	This meta-class allows for the definition of mappings of elements of a composite data type.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
firstElement	<code>SubElementRef</code>	0..1	aggr	This represents the first element referenced in the scope of the mapping. <b>Stereotypes:</b> <code>atpVariation</code> <b>Tags:</b> <code>vh.latestBindingTime=preCompileTime</code>
secondElement	<code>SubElementRef</code>	0..1	aggr	This represents the second element referenced in the scope of the mapping. <b>Stereotypes:</b> <code>atpVariation</code> <b>Tags:</b> <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 4.37: SubElementMapping**

<b>Class</b>	<b>SubElementRef</b> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
<b>Note</b>	This meta-class provides the ability to reference elements of composite data type.





<b>Class</b>	<b>SubElementRef</b> (abstract)			
<b>Base</b>	<i>ARObject</i>			
<b>Subclasses</b>	<a href="#">ApplicationCompositeDataTypeSubElementRef</a> , <a href="#">ImplementationDataTypeSubElementRef</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 4.38: SubElementRef**

<b>Class</b>	<b>ImplementationDataTypeSubElementRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	This meta-class represents the specialization of SubElementMapping with respect to ImplementationDataTypes.			
<b>Base</b>	<i>ARObject</i> , <i>SubElementRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
implementationDataTypeElement	<a href="#">ArVariableInImplementationDataInstanceRef</a>	0..1	aggr	This represents the referenced implementationDataTypeElement.
parameterImplementationDataTypeElement	<a href="#">ArParameterInImplementationDataInstanceRef</a>	0..1	aggr	This represents the referenced ImplementationDataTypeElement.

**Table 4.39: ImplementationDataTypeSubElementRef**

<b>Class</b>	<b>ApplicationCompositeDataTypeSubElementRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	This meta-class represents the specialization of SubElementMapping with respect to ApplicationCompositeDataTypes.			
<b>Base</b>	<i>ARObject</i> , <i>SubElementRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
applicationCompositeElement	<a href="#">ApplicationCompositeElementDataPrototype</a>	0..1	iref	This represents the referenced ApplicationCompositeDataPrototype. <b>InstanceRef implemented by:</b> <a href="#">ApplicationCompositeElementInPortInterfaceInstanceRef</a>

**Table 4.40: ApplicationCompositeDataTypeSubElementRef**

**[constr\_1883] Existence of [ApplicationCompositeDataTypeSubElementRef.applicationCompositeElement](#)** [For each [ApplicationCompositeDataTypeSubElementRef](#), the reference [applicationCompositeElement](#) shall exist **at the time when the RTE is generated.**.]()

**[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\_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 `rootVariableDataPrototype` at any time in the workflow.]()

[constr\_1518] Consistency of data types in the context of **ArParameterInImplementationDataInstanceRef** [The definition of attributes `contextDataPrototype` and `targetDataPrototype` shall be enclosed in the context of the definition of the data type used to type `rootParameterDataPrototype` at any time in the workflow.]()

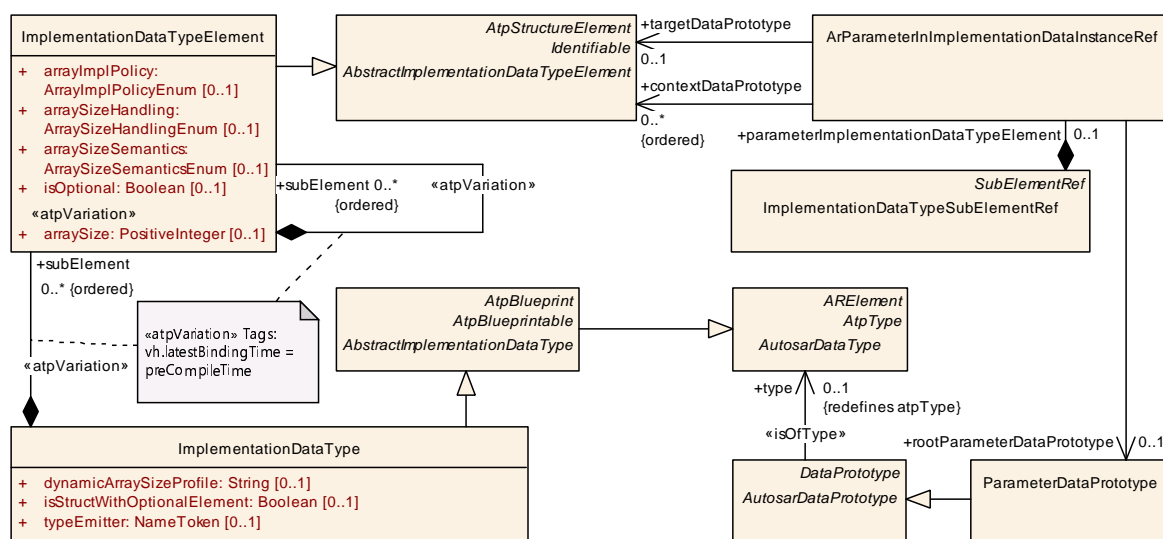


Figure 4.20: Implementation of the InstanceRef for the mapping of elements of a **ParameterDataPrototype** typed by a composite implementation data type

### 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`,
- `BITFIELD_TEXTTABLE`, and
- `RAT_FUNC` - as long as the semantics of the latter comes down to a reciprocal linear data scaling.

](RS\_SWCT\_03210)

**[TPS\_SWCT\_01561] Application of data conversion to composite Autosar-DataTypes** [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 to 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:

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

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:

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

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 – at the time when the RTE is generated – 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 of 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 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**<sup>7</sup> is represented by exactly one `TextTableValuePair.firstValue` (`[TPS_SWCT_01163]`) or `TextTableValuePair.secondValue` (`[TPS_SWCT_01164]`).

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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`.

---

<sup>7</sup>Depending 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]`.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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.

This rule shall be imposed **at the time when the RTE is generated.**]()

Example for [constr\_1307]: For a bit mask 0b00001000 only the corresponding values 8 and 0 are allowed.

<b>Class</b>	<b>TextTableMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Defines the mapping of two DataPrototypes typed by AutosarDataTypes that refer to CompuMethods of category TEXTTABLE, SCALE_LINEAR_AND_TEXTTABLE or BITFIELD_TEXTTABLE.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
bitfieldTextTableMaskFirst	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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
bitfieldTextTableMaskSecond	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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
identicalMapping	Boolean	0..1	attr	If identicalMapping is set == true the values of the two referenced DataPrototypes do not need any conversion of the values.
mappingDirection	MappingDirectionEnum	0..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 4.41: TextTableMapping**

**[constr\_1884] Existence of attribute `TextTableMapping.identicalMapping`** [For each `TextTableMapping`, the attribute `identicalMapping` shall exist **at the time when the RTE is generated.**]()

**[constr\_1885] Existence of attribute `TextTableMapping.mappingDirection`** [For each `TextTableMapping`, the attribute `mappingDirection` shall exist **at the time when the RTE is generated.**]()

<b>Enumeration</b>	<b>MappingDirectionEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
<b>Note</b>	Specifies the conversion direction for which the mapping is applicable.
<b>Literal</b>	<b>Description</b>
bidirectional	The <code>TextTableMapping</code> is applicable in both directions. <b>Tags:</b> atp.EnumerationLiteralIndex=0
firstToSecond	The <code>TextTableMapping</code> is applicable in the direction from <code>firstDataPrototype / firstOperationArgument</code> referring into the <code>PortInterface</code> of the <code>PPortPrototype</code> to <code>secondDataPrototype / secondOperationArgument</code> referring into the <code>PortInterface</code> of the <code>RPortPrototype</code> . <b>Tags:</b> atp.EnumerationLiteralIndex=1
secondToFirst	The <code>TextTableMapping</code> is applicable in the direction from <code>secondDataPrototype / secondOperationArgument</code> referring into the <code>PortInterface</code> of the <code>PPortPrototype</code> to <code>firstDataPrototype / firstOperationArgument</code> referring into the <code>PortInterface</code> of the <code>RPortPrototype</code> . <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 4.42: MappingDirectionEnum**

<b>Class</b>	<b>TextTableValuePair</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Defines a pair of text values which are translated into each other.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
firstValue	Numerical	0..1	attr	Value of first <code>DataPrototype</code> provided similar to a numerical <code>ValueSpecification</code> which is intended to be assigned to a <code>Primitive</code> data element. Note that the numerical value is a variant, it can be computed by a formula. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
secondValue	Numerical	0..1	attr	Value of second <code>DataPrototype</code> provided similar to a numerical <code>ValueSpecification</code> which is intended to be assigned to a <code>Primitive</code> data element. Note that the numerical value is a variant, it can be computed by a formula. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 4.43: TextTableValuePair**

**[constr\_1886] Existence of attribute `TextTableValuePair.firstValue`** [For each `TextTableValuePair`, the attribute `firstValue` shall exist **at the time when the RTE is generated.**]()

[constr\_1887] Existence of attribute `TextTableValuePair.secondValue` [For each `TextTableValuePair`, the attribute `secondValue` shall exist at the time when the RTE is generated.]()

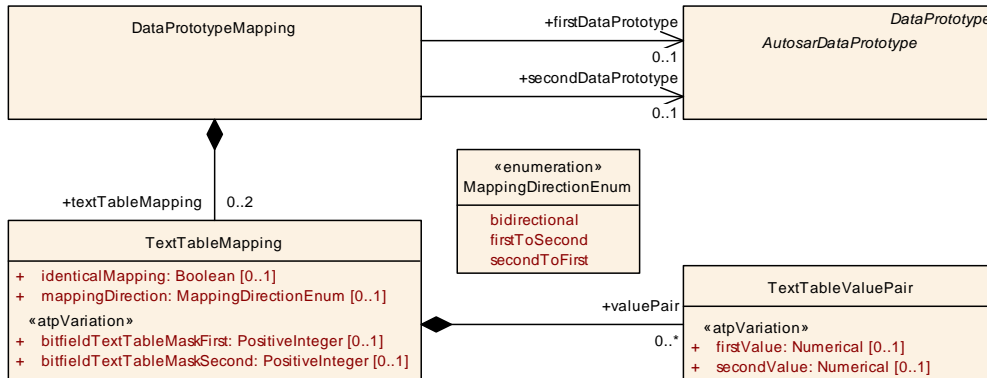


Figure 4.21: Mapping of DataPrototypes that eventually refer to `CompuMethods` of category `TEXTTABLE`, `SCALE_LINEAR_AND_TEXTTABLE`, and `BITFIELD_TEXTTABLE`

### 4.3.3 Relevance for Data Transformation

One (prominent) use-case for item 4 in [TPS\_SWCT\_01158] is the interaction between the `NvBlockSwComponentType` and the AUTOSAR Dcm.

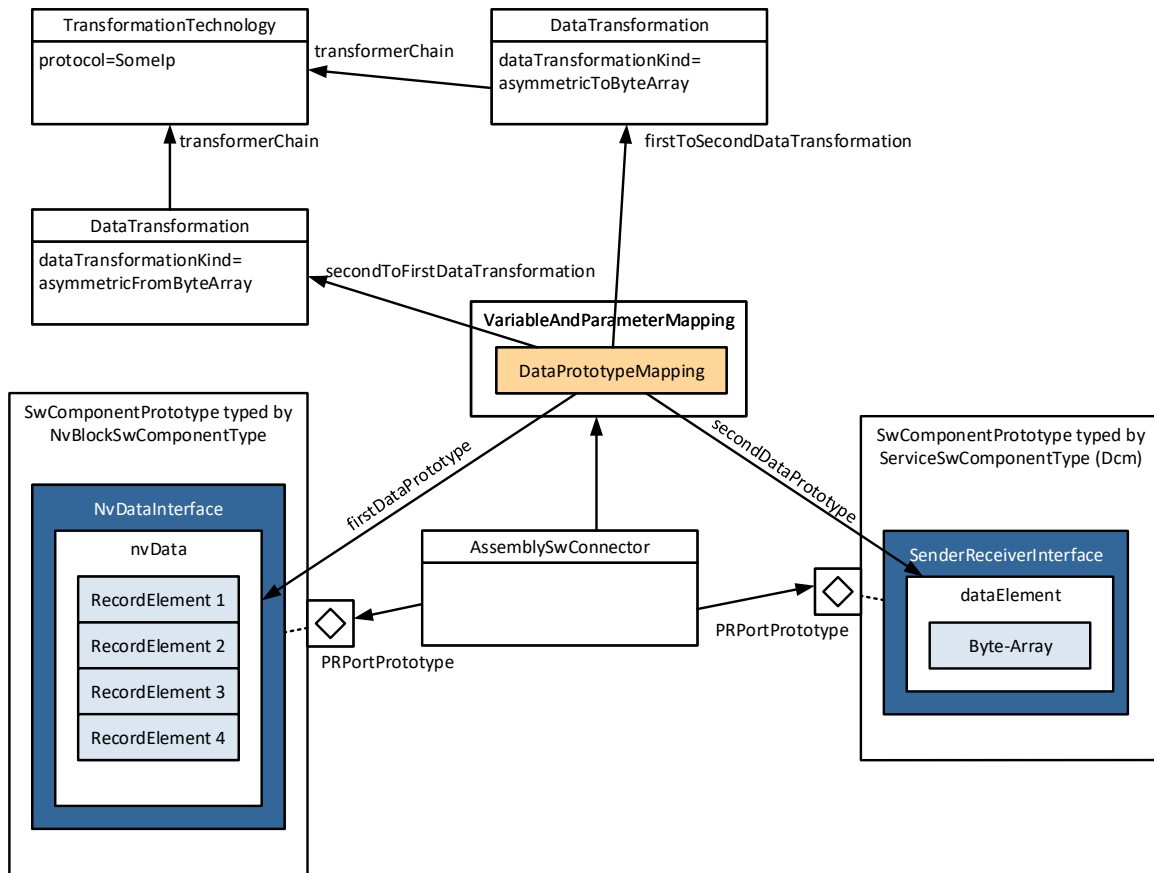
Specifically, the RTE will call a data transformer to convert the `uint8`-array representation of the diagnostic data available from a `PortPrototype` owned by the Dcm `ServiceSwComponentType` to a `VariableDataPrototype` owned by a `PortPrototype` of `NvBlockSwComponentType`.

For the configuration of this purpose, the applicable `DataPrototypeMapping` refers to a `DataTransformation` in the role `firstToSecondDataTransformation` and - for the case of two connected `PortPrototypes` that use asymmetric data transformation - `secondToFirstDataTransformation` (see Figure 4.22).



Figure 4.22: Configuration of Ecu-internal data transformation

Note that for this specific interaction between an `ApplicationSwComponentType` and a `ServiceSwComponentType` [TPS\_SWCT\_01579]/[TPS\_SWCT\_01831] applies which defines that attribute `isService` shall be set to false for the `dataElements` in `PortPrototypes` typed by a `SenderReceiverInterface`.



**Figure 4.23: Use case for the existence of asymmetric data transformation in both directions**

[TPS\_SWCT\_01768] Semantics of `DataPrototypeMapping.secondToFirstDataTransformation` [For symmetric data transformations (i.e. the value of attribute `DataTransformation.dataTransformationKind` is set to `DataTransformationKindEnum.symmetric`) it is sufficient to specify the reference `firstToSecondDataTransformation`.

There are, however, use cases for asymmetric data transformations between two connected `PRPortPrototype`s and in this case it is necessary to specify each direction separately.

For this purpose, the reference `secondToFirstDataTransformation` exists in addition to `firstToSecondDataTransformation`.] (*RS\_SWCT\_03210*)

Figure 4.23 describes the most prominent use case for the necessity to specify both `firstToSecondDataTransformation` and `secondToFirstDataTransformation`.

An `SwComponentPrototype` typed by `NvBlockSwComponentType` exposes a `PRPortPrototype` that is connected to another `PRPortPrototype` attached to an `SwComponentPrototype` that represents the Dcm service software-component.

The `PRPortPrototype` on the side of the `NvBlockSwComponentType` is typed by an `NvDataInterface` that in turn aggregates a single `nvData`. The data type used to define the `nvData` is a structured data type.

The service software-component representing the Dcm, however, is not capable of dealing with structured data types. It can only handle primitive types and arrays of primitive types, e.g. bytes.

Therefore, the existence of (asymmetric) data transformers is conveniently utilized to serialize the content of the structured data type into a linear array and vice versa.

To expressly define this intended semantics, the `DataPrototypeMapping` defines two references:

- `firstToSecondDataTransformation` that refers to a `DataTransformation` where attribute `dataTransformationKind` is set to the value `asymmetricToByteArray`. This reference represents the direction from the `NvBlockSwComponentType` to the Dcm.
- `secondToFirstDataTransformation` that refers to a `DataTransformation` where attribute `dataTransformationKind` is set to the value `asymmetricFromByteArray`. This reference represents the direction from the Dcm to the `NvBlockSwComponentType`.

This approach to modeling is formalized in [\[constr\\_1631\]](#) and [\[constr\\_1632\]](#).

**[constr\_1631] Applicability of `DataPrototypeMapping.secondToFirstDataTransformation`** [The reference to `DataTransformation` in the role `DataPrototypeMapping.secondToFirstDataTransformation` shall only exist if reference `DataPrototypeMapping.firstToSecondDataTransformation` exists and refers to a `DataTransformation` where attribute `dataTransformationKind` exists and is **not** set to the value `symmetric`.

This rule shall be imposed **at the time when the RTE is generated.** ]()

**[constr\_1632] Restriction for `firstToSecondDataTransformation` and `secondToFirstDataTransformation`** [If – **at the time when the RTE is generated** – both the reference `firstToSecondDataTransformation` and the reference `secondToFirstDataTransformation` exist in the context of the same `DataPrototypeMapping` then

- the `firstToSecondDataTransformation` shall refer to a `DataTransformation` with attribute `dataTransformationKind` set to `asymmetricToByteArray` and
- the `secondToFirstDataTransformation` shall refer to a `DataTransformation` with attribute `dataTransformationKind` set to `asymmetricFromByteArray`.

]()

<b>Class</b>	<b>DataTransformation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Transformer			
<b>Note</b>	A DataTransformation represents a transformer chain. It is an ordered list of transformers.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
data Transformation Kind	<a href="#">DataTransformationKindEnum</a>	0..1	attr	This attribute controls the kind of DataTransformation to be applied.
executeDespiteDataUnavailability	Boolean	1	attr	Specifies whether the transformer chain is executed even if no input data are available.
transformer Chain (ordered)	<a href="#">TransformationTechnology</a>	1..*	ref	This attribute represents the definition of a chain of transformers that are supposed to be executed according to the order of being referenced from DataTransformation.

**Table 4.44: DataTransformation**

**[constr\_1888] Existence of attribute [DataTransformation.executeDespiteDataUnavailability](#)** [For each [DataTransformation](#), the attribute [executeDespiteDataUnavailability](#) shall exist **at the time when the RTE is generated.**]([\(\)](#))

<b>Enumeration</b>	<b>DataTransformationKindEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Transformer
<b>Note</b>	This enumeration contributes to the definition of the scope of the DataTransformation.
<b>Literal</b>	<b>Description</b>
asymmetricFromByteArray	The DataTransformation shall only be applied to the receiving end only, i.e. transform from byte array to data type. <b>Tags:</b> atp.EnumerationLiteralIndex=0
asymmetricToByteArray	The DataTransformation shall be applied to the sending end only, i.e. from data type to byte array. <b>Tags:</b> atp.EnumerationLiteralIndex=1
symmetric	The DataTransformation shall be applied at both the sending and the receiving end of the communication. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 4.45: DataTransformationKindEnum**

## 4.4 Port Annotation

### 4.4.1 Introduction

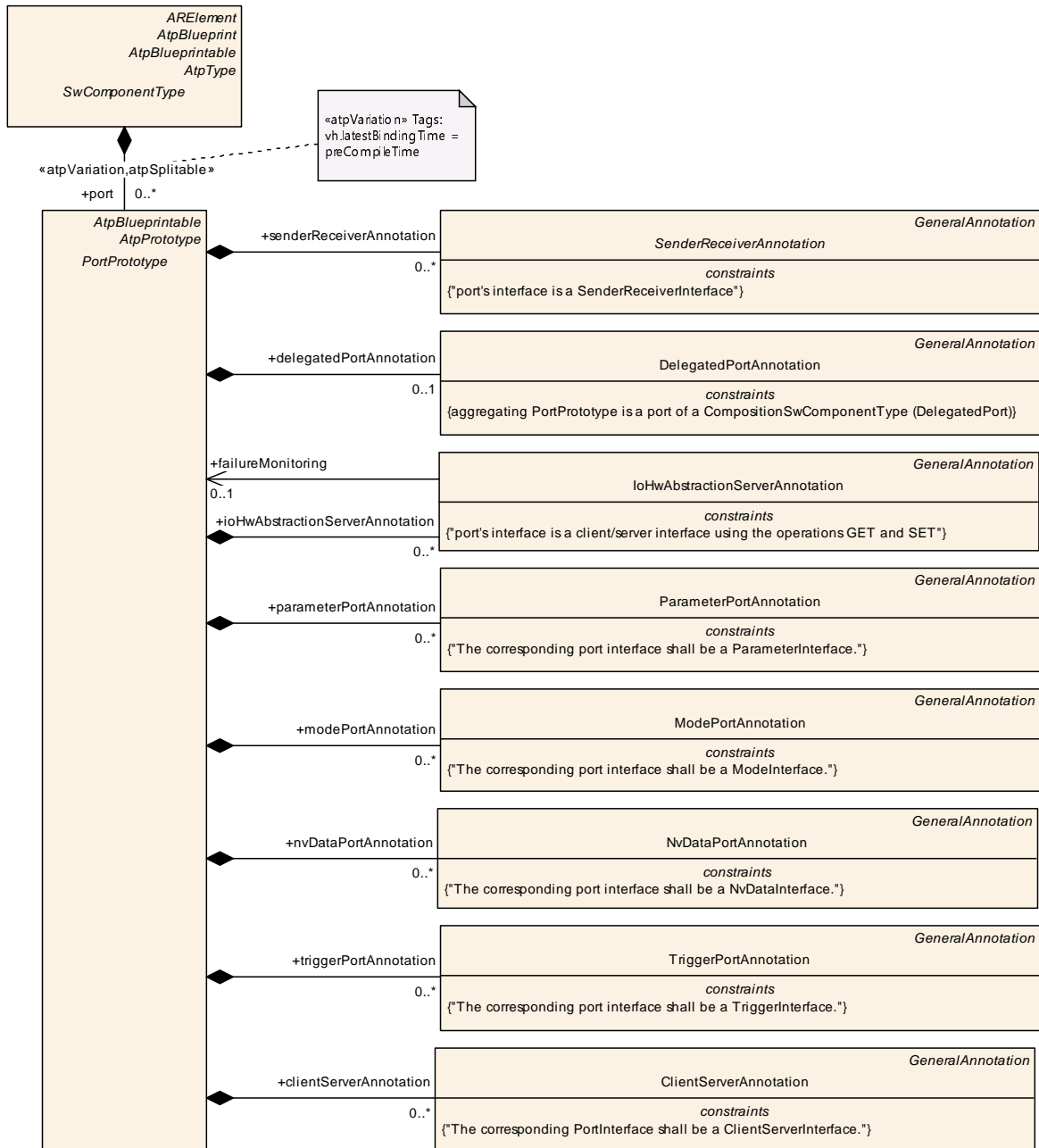
**[TPS\_SWCT\_01203] [PortPrototype](#) may own port annotations** [In addition to the formal specification required to implement the communication via ports, a [PortPrototype](#) may own so-called port annotations.

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.]([RS\\_SWCT\\_02110](#))



A summary of port-level annotations can be found in Figure 4.24.

[TPS\_SWCT\_01204] **GeneralAnnotation** [Beside formally specified attributes it is also possible to place textual information as provided in **GeneralAnnotation**.] ([RS\\_SWCT\\_02110](#))



**Figure 4.24: 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`.] ([RS\\_SWCT\\_02110](#))

The relationship of abstract meta-class `SenderReceiverAnnotation` to `SenderAnnotation` and `ReceiverAnnotation` is depicted in Figure 4.25.

<b>Class</b>	<b>SenderReceiverAnnotation</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	Annotation of the data elements in a port that realizes a sender/receiver interface.			
<b>Base</b>	ARObject, <a href="#">GeneralAnnotation</a>			
<b>Subclasses</b>	<a href="#">ReceiverAnnotation</a> , <a href="#">SenderAnnotation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
computed	Boolean	0..1	attr	Flag whether this data element was not measured directly but instead was calculated from possibly several other measured or calculated values.
dataElement	<a href="#">VariableDataPrototype</a>	0..1	ref	The instance of VariableDataPrototype annotated.
limitKind	<a href="#">DataLimitKindEnum</a>	0..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 AtomicSw ComponentType is transmitted to another AtomicSw ComponentType 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	<a href="#">ProcessingKindEnum</a>	0..1	attr	This attribute controls how data is processed according to the possible values of ProcessingKindEnum.

**Table 4.46: SenderReceiverAnnotation**

<b>Class</b>	<b>SenderAnnotation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	Annotation of a sender port, specifying properties of data elements that don't affect communication or generation of the RTE.			
<b>Base</b>	ARObject, <a href="#">GeneralAnnotation</a> , <a href="#">SenderReceiverAnnotation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 4.47: SenderAnnotation**

<b>Class</b>	<b>ReceiverAnnotation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">GeneralAnnotation</a> , <a href="#">SenderReceiverAnnotation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
signalAge	<a href="#">MultidimensionalTime</a>	0..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 4.48: ReceiverAnnotation**

<b>Enumeration</b>	<b>ProcessingKindEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
<b>Note</b>	Kind of processing which has been applied to a data element.
<b>Literal</b>	<b>Description</b>





Enumeration	ProcessingKindEnum
filtered	Indicates that a raw signal has been manipulated by some application software components by using filters. <b>Tags:</b> atp.EnumerationLiteralIndex=0
none	Indicates that none of the other option apply. <b>Tags:</b> atp.EnumerationLiteralIndex=1
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. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 4.49: ProcessingKindEnum**

Enumeration	DataLimitKindEnum
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
<b>Note</b>	Indicates whether the data element carries a minimum or maximum value, thereby limiting the current range of another value.
<b>Literal</b>	<b>Description</b>
max	Limitation to maximum value <b>Tags:</b> atp.EnumerationLiteralIndex=0
min	Limitation to minimum value <b>Tags:</b> atp.EnumerationLiteralIndex=1
none	No limitation applicable <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 4.50: 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.] ([RS\\_SWCT\\_02110](#))

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](#).

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.] ([RS\\_SWCT\\_02110](#))

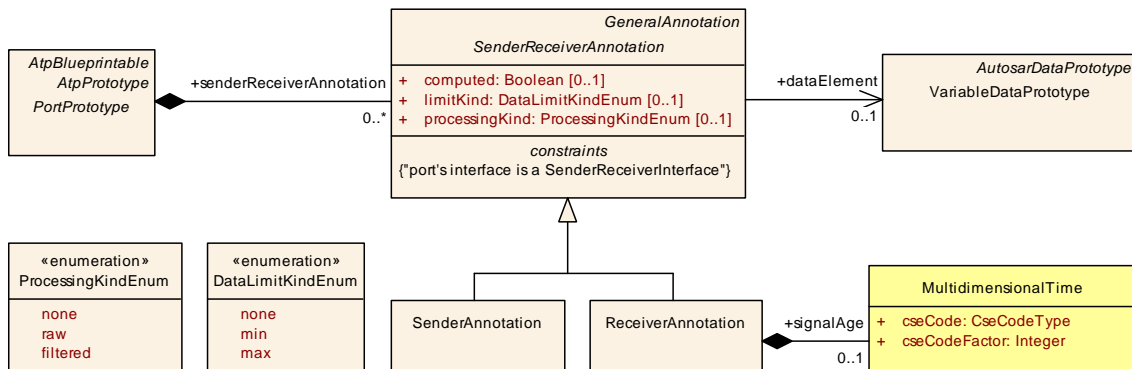


Figure 4.25: **SenderReceiverAnnotation**

[TPS\_SWCT\_01208] Grouping for **SenderReceiverAnnotation** [The **SenderReceiverAnnotation** for sender/receiver communication are grouped into

- processing type, indicating to some extent 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.] (*RS\_SWCT\_02110*)

For more information about meta-class **SenderReceiverAnnotation** please refer to Figure 4.25.

[constr\_4004] Context of **SenderReceiverAnnotation** [A **SenderReceiverAnnotation** shall only be aggregated by a **PortPrototype** typed by a **SenderReceiverInterface** at any time in the workflow.] ()

### 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**.] (*RS\_SWCT\_02110*)

Class	ClientServerAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation to a port regarding a certain Operation.			
Base	ARObject, GeneralAnnotation			
Attribute	Type	Mult.	Kind	Note
operation	ClientServerOperation	0..1	ref	This represents the ClientServerOperation that the Client ServerAnnotation corresponds to.

Table 4.51: **ClientServerAnnotation**

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

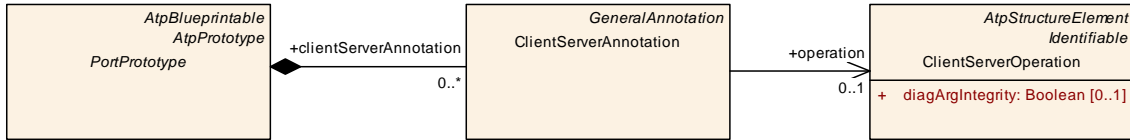


Figure 4.26: `ClientServerAnnotation`

**[constr\_4005] Context of `ClientServerAnnotation`** [A `ClientServerAnnotation` shall only be aggregated by a `PortPrototype` typed by a `ClientServerInterface` at any time in the workflow.]()

#### 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`.] (*RS\_SWCT\_02110*)

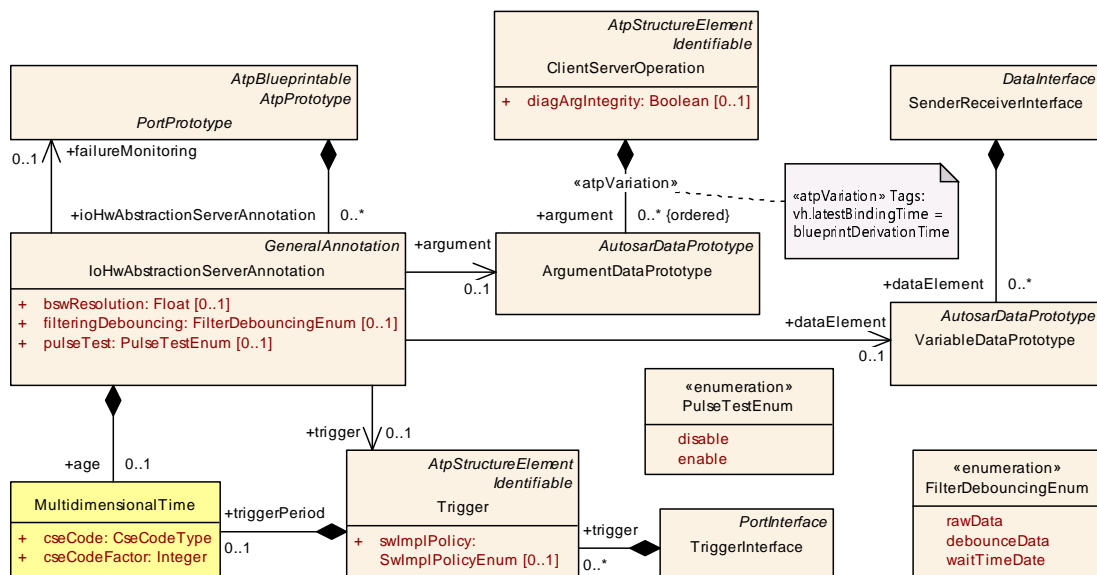
<b>Class</b>	<code>IoHwAbstractionServerAnnotation</code>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	The <code>IoHwAbstractionServerAnnotation</code> will only be used from a sensor- or an actuator component while interacting with the <code>IoHwAbstraction</code> layer.  Note that the "server" in the name of this meta-class is not meant to restrict the usage to <code>ClientServer</code> Interfaces.			
<b>Base</b>	<code>ARObject</code> , <code>GeneralAnnotation</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
age	<code>MultidimensionalTime</code>	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 <code>IoHwAbstraction</code> Layer <b>Tags:</b> xml.sequenceOffset=10
argument	<code>ArgumentDataPrototype</code>	0..1	ref	Reference to the corresponding <code>ArgumentDataPrototype</code> . <b>Tags:</b> xml.sequenceOffset=20
bswResolution	Float	0..1	attr	This value is determined by an appropriate combination of the range, the unit as well as the data-elements type, i.e. $(ecuSignalRange.upperLimit - ecuSignalRange.lowerLimit) / (2^{datatypelength} - 1)$ <b>Tags:</b> xml.sequenceOffset=30
dataElement	<code>VariableDataPrototype</code>	0..1	ref	Reference to the corresponding <code>VariableDataPrototype</code> . <b>Tags:</b> xml.sequenceOffset=40





Class	IoHwAbstractionServerAnnotation			
failure Monitoring	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 a 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.  <b>Tags:</b> xml.sequenceOffset=50
filtering Debouncing	FilterDebouncingEnum	0..1	attr	This attribute is used to indicate what kind of filtering/ debouncing has been put to the signal in the IoHw Abstraction 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  <b>Tags:</b> xml.sequenceOffset=60
pulseTest	PulseTestEnum	0..1	attr	This attribute indicates to the connected SensorActuator SwComponentType whether the VariableDataPrototype can be used to generate pulse test sequences using the IoHwAbstraction layer  <b>Tags:</b> xml.sequenceOffset=70
trigger	Trigger	0..1	ref	Reference to the corresponding Trigger.  <b>Tags:</b> xml.sequenceOffset=80

**Table 4.52: IoHwAbstractionServerAnnotation**



**Figure 4.27: IoHwAbstractionServerAnnotation**

<b>Enumeration</b>	<b>FilterDebouncingEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
<b>Note</b>	This enumeration defines possible values for the filter debouncing strategy.
<b>Literal</b>	<b>Description</b>
debounceData	The signal is a mean value <b>Tags:</b> atp.EnumerationLiteralIndex=0
rawData	Means that no modification of the signal has been applied. This is the default value <b>Tags:</b> atp.EnumerationLiteralIndex=1
waitTimeDate	The signal is delivered by a GET operation after a certain amount of time <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 4.53: FilterDebouncingEnum**

<b>Enumeration</b>	<b>PulseTestEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
<b>Note</b>	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
<b>Literal</b>	<b>Description</b>
disable	Disables the pulse test <b>Tags:</b> atp.EnumerationLiteralIndex=0
enable	Enables the pulse test <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 4.54: PulseTestEnum**

**[TPS\_SWCT\_01211] Assign several annotations to [ArgumentDataPrototype](#)** [The [ClientServerOperations](#) provide an [ArgumentDataPrototype](#) where several annotations can be assigned to.] ([RS\\_SWCT\\_02110](#))

They are depicted in the [IoHwAbstractionServerAnnotation](#) meta-class in Figure 4.27.

A detailed description of the attributes can be found in the IoHwAbstraction Layer software specification document [16].

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](#).

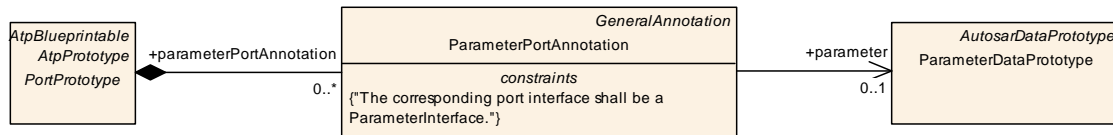


The data provided at the `PortPrototype` is calibration parameters. The `ParameterPortAnnotation` provides a reference to a particular `ParameterDataPrototype`.] (*RS\_SWCT\_02110*)

<b>Class</b>	<b>ParameterPortAnnotation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	Annotation to a port used for calibration regarding a certain <code>ParameterDataPrototype</code> .			
<b>Base</b>	<i>ARObject</i> , <i>GeneralAnnotation</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
parameter	<a href="#">ParameterDataPrototype</a>	0..1	ref	The instance of annotated <code>ParameterDataPrototype</code> .

**Table 4.55: ParameterPortAnnotation**

The main use-case is to allow easy access to the information which calibration parameters influence the data on the `PortPrototype`.



**Figure 4.28: ParameterPortAnnotation**

**[constr\_4006] Context of `ParameterPortAnnotation`** [A `ParameterPortAnnotation` shall only be aggregated by a `PPortPrototype` owned by a `ParameterSwComponentType` at any time in the workflow.] ()

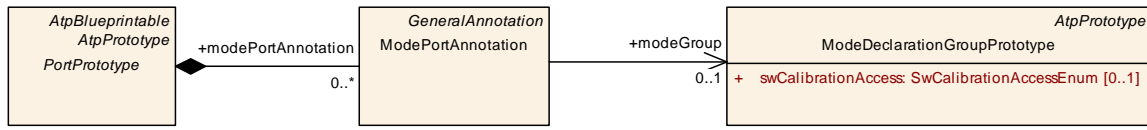
#### 4.4.6 Mode Port Annotation

**[TPS\_SWCT\_01213] `ModePortAnnotation`** [The `ModePortAnnotation` can be used to provide more information with respect to the mode declaration group prototype of the `PortPrototype`.] ()

<b>Class</b>	<b>ModePortAnnotation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	Annotation to a port used for calibration regarding a certain <code>ModeDeclarationGroupPrototype</code> .			
<b>Base</b>	<i>ARObject</i> , <i>GeneralAnnotation</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
modeGroup	<a href="#">ModeDeclarationGroupPrototype</a>	0..1	ref	The instance of annotated <code>ModeDeclarationGroupPrototype</code> .

**Table 4.56: ModePortAnnotation**

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



**Figure 4.29: ModePortAnnotation**

**[constr\_4007] Context of ModePortAnnotation** [A ModePortAnnotation shall only be aggregated by a PortPrototype typed by a ModeSwitchInterface at any time in the workflow.]()

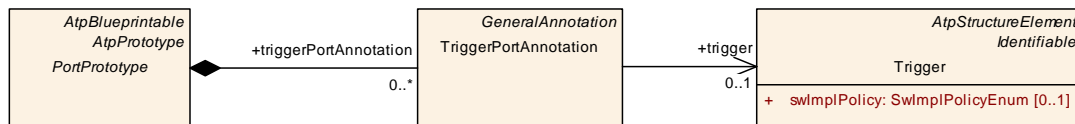
#### 4.4.7 Trigger Port Annotation

**[TPS\_SWCT\_01214] TriggerPortAnnotation** [The TriggerPortAnnotation can be used to provide more information with respect to the Trigger of the PortPrototype.](RS\_SWCT\_02110)

<b>Class</b>	TriggerPortAnnotation			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	Annotation to a port used for calibration regarding a certain Trigger.			
<b>Base</b>	ARObject, GeneralAnnotation			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
trigger	Trigger	0..1	ref	The instance of annotated trigger.

**Table 4.57: TriggerPortAnnotation**

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



**Figure 4.30: TriggerPortAnnotation**

**[constr\_4008] Context of TriggerPortAnnotation** [A TriggerPortAnnotation shall only be aggregated by a PortPrototype typed by a TriggerInterface at any time in the workflow.]()

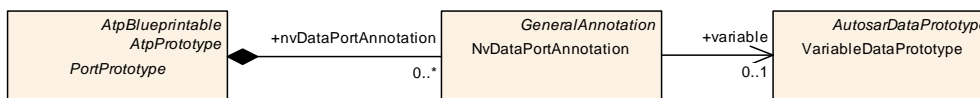
#### 4.4.8 Non Volatile Data Port Annotation

**[TPS\_SWCT\_01215] NvDataPortAnnotation** [The NvDataPortAnnotation can be used to provide more information with respect to the non-volatile data of the PortPrototype.](RS\_SWCT\_02110)

<b>Class</b>	<b>NvDataPortAnnotation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	Annotation to a port regarding a certain VariableDataPrototype.			
<b>Base</b>	ARObject, <a href="#">GeneralAnnotation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
variable	<a href="#">VariableDataPrototype</a>	0..1	ref	The instance of nv data annotated.

**Table 4.58: NvDataPortAnnotation**

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



**Figure 4.31: NvDataPortAnnotation**

**[constr\_4009] Context of [NvDataPortAnnotation](#)** [An [NvDataPortAnnotation](#) shall only be aggregated by a [PortPrototype](#) typed by an [NvDataInterface](#) at any time in the workflow.]()

#### 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 the [CompositionSwComponentType](#).] ([RS\\_SWCT\\_02110](#))

<b>Class</b>	<b>DelegatedPortAnnotation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
<b>Note</b>	Annotation to a "delegated port" to specify the Signal Fan In or Signal Fan Out inside the CompositionSwComponentType.			
<b>Base</b>	ARObject, <a href="#">GeneralAnnotation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
signalFan	<a href="#">SignalFanEnum</a>	0..1	attr	Specifies the Signal Fan In or Signal Fan Out inside the Composition Type.

**Table 4.59: DelegatedPortAnnotation**

<b>Enumeration</b>	<b>SignalFanEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
<b>Note</b>	Signal Fan inside the Composition Component Type.
<b>Literal</b>	<b>Description</b>
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. <b>Tags:</b> atp.EnumerationLiteralIndex=0
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. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 4.60: SignalFanEnum**

[TPS\_SWCT\_01217] Semantics of [DelegatedPortAnnotation.signalFan](#) [The attribute values have following definition:

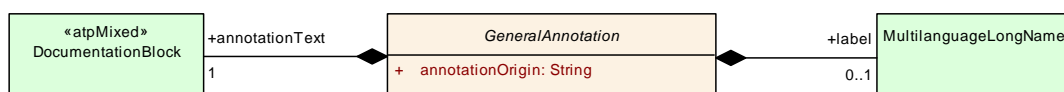
- **single:** the internal connections in the [CompositionSwComponentType](#) via [DelegationSwConnectors](#) and [AssemblySwConnectors](#) are defined in a way that each [dataElement](#) present in the [SenderReceiverInterfaces](#) or [operation](#) in the [ClientServerInterfaces](#) of the outer [PortPrototype](#) 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.

] ([RS\\_SWCT\\_02110](#))

[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 4.32 for an overview).



**Figure 4.32: textual information in annotations**

<b>Class</b>	<b>GeneralAnnotation</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::GeneralAnnotation			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i>			
<b>Subclasses</b>	<a href="#">Annotation</a> , <a href="#">ClientServerAnnotation</a> , <a href="#">DelegatedPortAnnotation</a> , <a href="#">IoHwAbstractionServerAnnotation</a> , <a href="#">ModePortAnnotation</a> , <a href="#">NvDataPortAnnotation</a> , <a href="#">ParameterPortAnnotation</a> , <a href="#">SenderReceiverAnnotation</a> , <a href="#">TriggerPortAnnotation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
annotation Origin	<a href="#">String</a>	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.  <b>Tags:</b> xml.sequenceOffset=30
annotationText	<a href="#">DocumentationBlock</a>	1	aggr	This is the text of the annotation.  <b>Tags:</b> xml.sequenceOffset=40
label	MultilanguageLong Name	0..1	aggr	This is the headline for the annotation.  <b>Tags:</b> xml.sequenceOffset=20

**Table 4.61: 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.

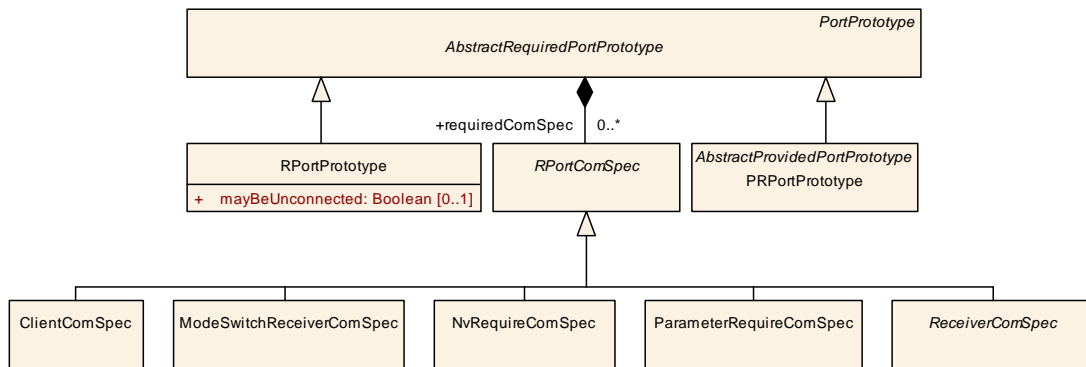
Such [PortInterface](#) however, only describes structure and does not include information about whether communication needs to be done reliably, or whether an initial value exists in case the real data is not yet available.

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. | ([RS\\_SWCT\\_02030](#))

Note that the communication specification is optional, i.e. its existence is not required in any case. Figures [4.33](#) and [4.34](#) provide an overview of communication specifications. The derived meta-classes are explained in the following sub-chapters.

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](#).



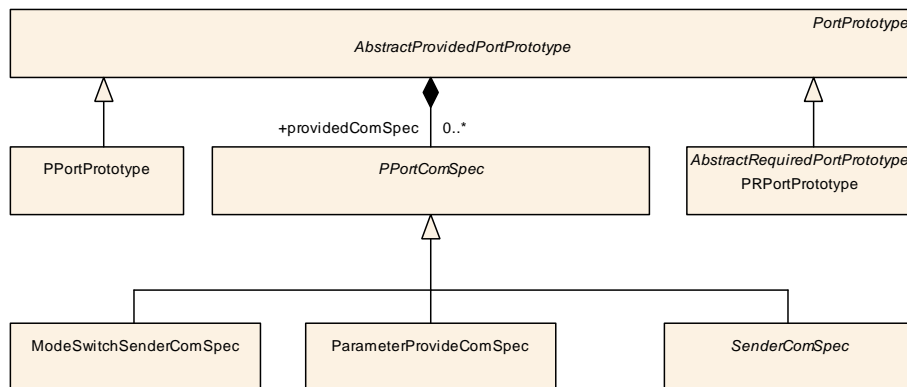
**Figure 4.33: Overview of communication attributes of RPortPrototype**

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 defining `initValues` in relation to the associated `DataPrototype`. For further details about the representation of `initValues` please refer to section 5.7.2.

Furthermore, [constr\_1043] applies such that only specific subclasses of `ComSpec` can be owned by `PortPrototypes` typed by the corresponding kind of `PortInterface`.



**Figure 4.34: Overview of communication attributes of PPortPrototype**

**[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 `operation` **at the time when the contract phase generation is executed.**]()

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 `operation`.

**[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 `operation` at the time when the contract phase generation is executed.]()

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 `operation`.

[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\_02030*, *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 `operation` at the time when the contract phase generation is executed.]()

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 `operation`.

In the same manner, it is not allowed that two or more `RPortComSpec` exist in the context of one `PRPortPrototype` that refer to the same `dataElement` or `operation`.

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.

<b>Class</b>	<i>PPortComSpec</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i>			
<b>Subclasses</b>	<i>ModeSwitchSenderComSpec</i> , <i>NvProvideComSpec</i> , <i>ParameterProvideComSpec</i> , <i>SenderComSpec</i> , <i>ServerComSpec</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 4.62: PPortComSpec**

<b>Class</b>	<i>RPortComSpec</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i>			





<b>Class</b>	<i>RPortComSpec</i> (abstract)			
<b>Subclasses</b>	<i>ClientComSpec</i> , <i>ModeSwitchReceiverComSpec</i> , <i>NvRequireComSpec</i> , <i>ParameterRequireComSpec</i> , <i>ReceiverComSpec</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 4.63: RPortComSpec**

**[constr\_1043] PortInterface vs. ComSpec** [The allowed combinations **at any time in the workflow** of a specific kind of *PortInterface* and a kind of *ComSpec* are documented in Table 4.64.]()

<b>PortInterface</b>	<b>ComSpec</b>
<i>SenderReceiverInterface</i>	<i>SenderComSpec</i> , <i>ReceiverComSpec</i>
<i>ClientServerInterface</i>	<i>ClientComSpec</i> , <i>ServerComSpec</i>
<i>ModeSwitchInterface</i>	<i>ModeSwitchSenderComSpec</i> , <i>ModeSwitchReceiverComSpec</i>
<i>ParameterInterface</i>	<i>ParameterProvideComSpec</i> , <i>ParameterRequireComSpec</i>
<i>NvDataInterface</i>	<i>NvRequireComSpec</i> , <i>NvProvideComSpec</i>

**Table 4.64: 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.

**[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 *Non-queuedSenderComSpec* or *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.]()

### 4.5.1.1 Receiver ComSpec

Figure 4.35 shows the meta-model of the communication attributes relevant sender-receiver communication at an `RPortPrototype`.

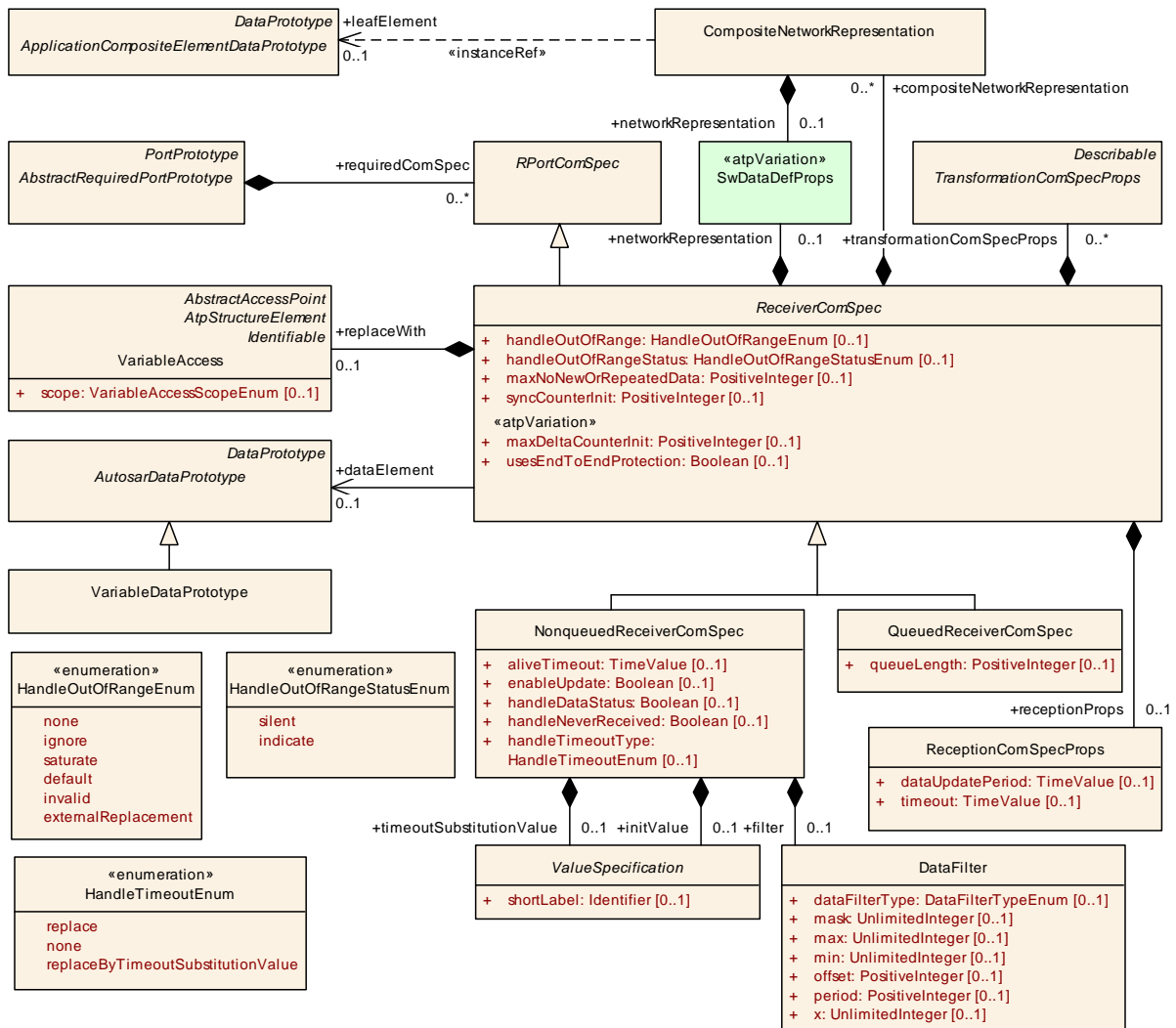


Figure 4.35: Communication attributes of `RPortPrototype` with respect to sender-receiver communication.

[constr\_1538] Restriction for `ReceiverComSpec.dataElement` [The reference `ReceiverComSpec.dataElement` shall not refer to an `ArgumentDataPrototype` or `ParameterDataPrototype` at any time in the workflow.]()

**[constr\_1103] NonqueuedReceiverComSpec and enableUpdate** [A `NonqueuedReceiverComSpec` that has the value of attribute `enableUpdate` set to `true` **at the time when the contract phase generation is executed** may not reference a `dataElement` that in turn is referenced by a `VariableAccess` in the role `dataReadAccess`.]()

In general, it is considered beneficial for software-components to define `initValues` for all the `dataElements` received by `RPortPrototypes`.

These `initValues` are required by the RTE for several functionalities, e.g. :

- Providing a default value for not yet received `dataElements` (see [TPS\_SWCT\_01220]).
- Providing default values in case of unconnected `RPortPrototypes` (see [constr\_1100]).
- Partial mapping of composite data (see [constr\_1280]).

Therefore, the availability of `initValue` increases the flexibility of the usage of the software-component in different scenarios.

On the other hand, there are also use cases where `initValues` are not mandatory, i.e. the `DataPrototype` remains intentionally uninitialized. This is expressed by applying a `SwAddrMethod` where the `sectionInitializationPolicy` is set to `NO-INIT`, or when the software component is intentionally only prepared for intra-partition communication.

In response to these conflicting objectives [TPS\_SWCT\_01688] is written as a recommendation as opposed to a binding constraint.

**[TPS\_SWCT\_01688] `initValue` should exist in an `RPortPrototype`** [The optional attribute `initValue` should exist if the enclosing `NonqueuedReceiverComSpec` is owned by an `RPortPrototype`.]()

**[TPS\_SWCT\_01455] Duplicate existence of `initValue` in the context of a `PRPortPrototype`** [If an `initValue` is defined in a `NonqueuedReceiverComSpec` owned by a `PRPortPrototype`, its value shall be ignored.](RS\_SWCT\_03250)

**[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.]()

**[constr\_1891] Existence of attribute `NonqueuedReceiverComSpec.initValue`** [For each `NonqueuedReceiverComSpec`, attribute `initValue` shall exist **at the time when the contract phase generation is executed**.]()

<b>Class</b>	<b>ReceiverComSpec</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Receiver-specific communication attributes (RPortPrototype typed by SenderReceiverInterface).			
<b>Base</b>	ARObject, RPortComSpec			
<b>Subclasses</b>	NonqueuedReceiverComSpec, QueuedReceiverComSpec			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
composite Network Representation	<a href="#">CompositeNetworkRepresentation</a>	*	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 Application CompositeDataTypes.
dataElement	<a href="#">AutosarDataPrototype</a>	0..1	ref	Data element these attributes belong to.
handleOutOfRange	<a href="#">HandleOutOfRangeEnum</a>	0..1	attr	This attribute controls how values that are out of the specified range are handled according to the values of HandleOutOfRangeEnum.
handleOutOfRangeStatus	<a href="#">HandleOutOfRangeStatusEnum</a>	0..1	attr	Control the way how return values are created in case of an out-of-range situation.
maxDeltaCounterInit	PositiveInteger	0..1	attr	<p>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 MaxDeltaCounter Init is 1, then at the next reception the receiver can accept Counters with values 2 and 3, but not 4.</p> <p>Note that if the receiver does not receive new Data at a consecutive read, then the receiver increments the tolerance by 1.</p> <p>Caveat: The E2E wrapper approach involves technologies that are not subjected to the AUTOSAR standard and is superseded by the superior E2E transformer approach (which is fully standardized by AUTOSAR). Hence, new projects (without legacy constraints due to carry-over parts) shall use the fully standardized E2E transformer approach.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
maxNoNewOrRepeatedData	PositiveInteger	0..1	attr	<p>The maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions.</p> <p>Caveat: The E2E wrapper approach involves technologies that are not subjected to the AUTOSAR standard and is superseded by the superior E2E transformer approach (which is fully standardized by AUTOSAR). Hence, new projects (without legacy constraints due to carry-over parts) shall use the fully standardized E2E transformer approach.</p>
networkRepresentation	<a href="#">SwDataDefProps</a>	0..1	aggr	A networkRepresentation is used to define how the data Element is mapped to a communication bus.
receptionProps	<a href="#">ReceptionComSpecProps</a>	0..1	aggr	"This aggregation represents the definition transmission props in the context of the enclosing ReceiverComSpec.
replaceWith	<a href="#">VariableAccess</a>	0..1	aggr	This aggregation is used to identify the AutosarData Prototype to be taken for sourcing an external replacement in the out-of-range handling.





<b>Class</b>	<b>ReceiverComSpec</b> (abstract)			
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.  Caveat: The E2E wrapper approach involves technologies that are not subjected to the AUTOSAR standard and is superseded by the superior E2E transformer approach (which is fully standardized by AUTOSAR). Hence, new projects (without legacy constraints due to carry-over parts) shall use the fully standardized E2E transformer approach.
transformationComSpecProps	<a href="#">TransformationComSpecProps</a>	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.
usesEndToEndProtection	Boolean	0..1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection.  Caveat: The E2E wrapper approach involves technologies that are not subjected to the AUTOSAR standard and is superseded by the superior E2E transformer approach (which is fully standardized by AUTOSAR). Hence, new projects (without legacy constraints due to carry-over parts) shall use the fully standardized E2E transformer approach.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 4.65: ReceiverComSpec**

<b>Enumeration</b>	<b>HandleOutOfRangeStatusEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication
<b>Note</b>	This enumeration defines how the RTE handles values that are out of range.
<b>Literal</b>	<b>Description</b>
indicate	The RTE sets the return status to RTE_E_OUT_OF_RANGE if the received value is out of range and the attribute handleOutOfRange is not set to "none" or "invalid". <b>Tags:</b> atp.EnumerationLiteralIndex=0
silent	The RTE sets the return status to RTE_E_OK <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 4.66: HandleOutOfRangeStatusEnum**

<b>Class</b>	<b>NonqueuedReceiverComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes specific to non-queued receiving.			
<b>Base</b>	<i>ARObject</i> , <a href="#">RPortComSpec</a> , <a href="#">ReceiverComSpec</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
aliveTimeout	<a href="#">TimeValue</a>	0..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.





Class	NonqueuedReceiverComSpec			
enableUpdate	Boolean	0..1	attr	This attribute controls whether application code is entitled to check whether the value of the corresponding VariableDataPrototype has been updated.
filter	<a href="#">DataFilter</a>	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
handleData Status	Boolean	0..1	attr	If this attribute is set to true, then 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.
handleNever Received	Boolean	0..1	attr	<p>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.</p> <ul style="list-style-type: none"> <li>• If the value of this attribute is set to "true" the flag is required.</li> <li>• If set to "false", the RTE shall not support the "never received" functionality for the corresponding VariableDataPrototype.</li> </ul>
handleTimeout Type	<a href="#">HandleTimeoutEnum</a>	0..1	attr	This attribute controls the behavior with respect to the handling of timeouts.
initValue	<a href="#">ValueSpecification</a>	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, then the receiver's value will be used.
timeout Substitution Value	<a href="#">ValueSpecification</a>	0..1	aggr	This attribute represents the substitution value applicable in the case of a timeout.

**Table 4.67: NonqueuedReceiverComSpec**

Class	QueuedReceiverComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to queued receiving.			
Base	<a href="#">ARObject</a> , <a href="#">RPortComSpec</a> , <a href="#">ReceiverComSpec</a>			
Attribute	Type	Mult.	Kind	Note
queueLength	PositiveInteger	0..1	attr	Length of queue for received events.

**Table 4.68: QueuedReceiverComSpec**

**[constr\_1889] Existence of attribute [QueuedReceiverComSpec.queueLength](#)**  
 [For each [QueuedReceiverComSpec](#), attribute [queueLength](#) shall exist **at the time when the contract phase generation is executed.**] ()

Class	ReceptionComSpecProps			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	This meta-class defines a set of reception attributes which the application software is assumed to implement.			
Base	<a href="#">ARObject</a>			
Attribute	Type	Mult.	Kind	Note





Class	ReceptionComSpecProps			
dataUpdatePeriod	TimeValue	0..1	attr	This attribute defines the period in which the application shall check for updated data. This attribute is used for the configuration of the E2E protection, but may also indicate a general data reception period.
timeout	TimeValue	0..1	attr	This attribute defines the time interval after which the application shall assume that the to be received data reception has timed out, i.e. the respective data has not been received for that amount of time.

**Table 4.69: ReceptionComSpecProps**

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. <b>Tags:</b> atp.EnumerationLiteralIndex=0
replace	If set to replace, the replacement value shall be the ComInitValue. <b>Tags:</b> atp.EnumerationLiteralIndex=1
replaceByTimeoutSubstitutionValue	If set to replace, the replacement value shall be the timeout substitution value. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 4.70: 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. <b>Tags:</b> xml.xsd.customType=TIME-VALUE xml.xsd.type=double

**Table 4.71: TimeValue**

**[constr\_1129] swImplPolicy and NonqueuedReceiverComSpec** [The attribute `swImplPolicy` of a `dataElement` referenced by a `NonqueuedReceiverComSpec` shall not be set to the value **queued at the time when the contract phase generation is executed.**]()

**[constr\_1130] swImplPolicy and QueuedReceiverComSpec** [The attribute `swImplPolicy` of a `dataElement` referenced by a `QueuedReceiverComSpec` shall be set to the value **queued at the time when the contract phase generation is executed.**]()

**[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`.

This rule shall be applied **at the time when the contract phase generation is executed.**]()

**[TPS\_SWCT\_01753] Application of compatibility rules for `ReceiverComSpec.replaceWith`** [Compatibility rules as formulated by `[constr_1068]` and `[constr_1187]` shall be applicable for the reference `ReceiverComSpec.replaceWith.`]()

**[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 to take the `networkRepresentation` for data mapping is done in the context of the AUTOSAR System Template [10]. 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` (and vice versa) **at the time when the contract phase generation is executed.**]()

**[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 **at the time when the contract phase generation is executed.**]

For each such `compositeNetworkRepresentation`, attributes `leafElement` and `networkRepresentation` shall exist **at the time when the contract phase generation is executed.**]

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.

**[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.]

**[TPS\_SWCT\_01682] The meaning of E2E-related attributes in a `ReceiverComSpec` if a `TransformationComSpecProps` of type `EndToEndTransformationComSpecProps` is defined.** [The attributes `usesEndToEndProtection`, `syncCounterInit`, `maxDeltaCounterInit`, and `maxNoNewOrRepeatedData` in `ReceiverComSpec` have no meaning if a `TransformationComSpecProps` of type `EndToEndTransformationComSpecProps` is defined in the same `ReceiverComSpec`.]

#### 4.5.1.2 Sender ComSpec

The communication attributes on the sender side are sketched in Figure 4.36.

**[constr\_1131] `swImplPolicy` and `NonqueuedSenderComSpec`** [The attribute `swImplPolicy` of a `dataElement` referenced by a `NonqueuedSenderComSpec` shall not be set to the value `queued` **at the time when the contract phase generation is executed.**]

**[constr\_1132] `swImplPolicy` and `QueuedSenderComSpec`** [The attribute `swImplPolicy` of a `dataElement` referenced by a `QueuedSenderComSpec` shall be set to the value `queued` **at the time when the contract phase generation is executed.**]



[TPS\_SWCT\_01751] The meaning of E2E-related attributes in a **SenderComSpec** if a **TransformationComSpecProps** of type **EndToEndTransformationComSpecProps** is defined [The attribute `usesEndToEndProtection` has no meaning if a **TransformationComSpecProps** of type **EndToEndTransformationComSpecProps** is defined in the same **SenderComSpec**.]()

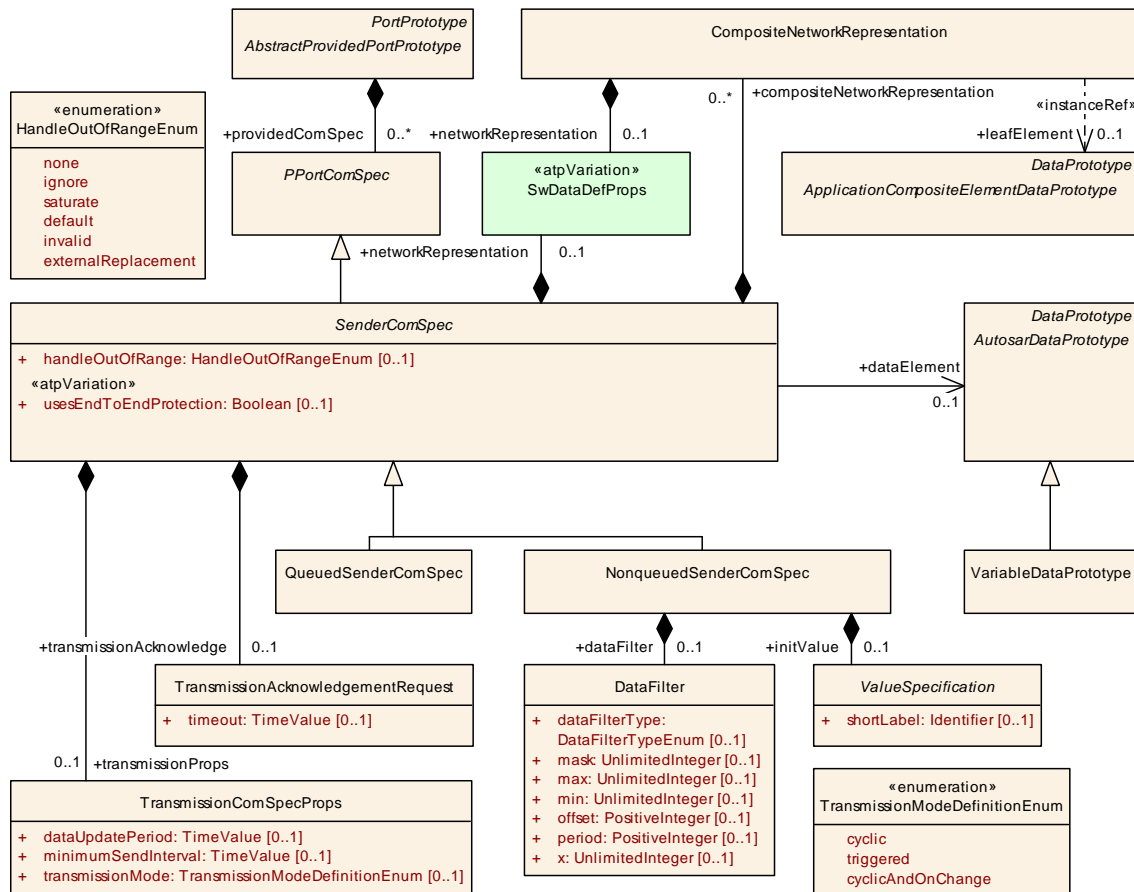


Figure 4.36: Communication attributes of **PPortPrototype** with respect to sender-receiver communication.

Please note:

- `SenderComSpec.usesEndToEndProtection` does not have any influence on code generation.  
It could be used, for example, by a validation framework to make sure that, if set to `True` the `dataElement` meets a transformer configuration for all respective `SwConnectors` connecting to the `PortPrototype` that owns the `SenderComSpec`.
- `SenderComSpec.usesEndToEndProtection` could be used as a statement from the application developer that the given `dataElement` shall be end-to-end protected.

However, it seems far-fetched for an application developer to expressly state that a `dataElement` shall **not** be end-to-end protected. This goes beyond the responsibility of an application developer.

Therefore, two relevant states for `SenderComSpec.usesEndToEndProtection` can be expected:

- attribute exists and is set to `True` (application developer asserts the necessity to end-to-end protect the `dataElement`)
- attribute does not exist (application developer doesn't care)
- The application developer may not have enough oversight to envision how the `dataElement` is communicated, i.e. local vs. network communication. Setting `usesEndToEndProtection` to `True` and then deploy the enclosing software-component such that it communicates only locally on the respective `PortPrototype` also seems unusual for the current situation regarding transformer-based communication.

**[constr\_1539] Restriction for `SenderComSpec.dataElement`** [The reference `SenderComSpec.dataElement` shall not refer to an `ArgumentDataPrototype` or `ParameterDataPrototype` at the time when the contract phase generation is executed.]()

<b>Class</b>	<i>SenderComSpec</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes for a sender port (PPortPrototype typed by SenderReceiverInterface).			
<b>Base</b>	ARObject, PPortComSpec			
<b>Subclasses</b>	NonqueuedSenderComSpec, QueuedSenderComSpec			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
composite Network Representation	CompositeNetworkRepresentation	*	aggr	This represents a CompositeNetworkRepresentation defined in the context of a SenderComSpec.
dataElement	AutosarDataPrototype	0..1	ref	Data element these quality of service attributes apply to.
handleOutOfRange	HandleOutOfRangeEnum	0..1	attr	This attribute controls how out-of-range values shall be dealt with.
network Representation	SwDataDefProps	0..1	aggr	A networkRepresentation is used to define how the data Element is mapped to a communication bus.
transmission Acknowledge	TransmissionAcknowledgementRequest	0..1	aggr	Requested transmission acknowledgement for data element.
transmission Props	TransmissionComSpecProps	0..1	aggr	This aggregation represents the definition transmission props in the context of the enclosing SenderComSpec.
usesEndToEndProtection	Boolean	0..1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 4.72: SenderComSpec**

[TPS\_SWCT\_01820] Existence of attribute [SenderComSpec.handleOutOfRange](#)  
[If attribute [SenderComSpec.handleOutOfRange](#) does not exist at the time when the RTE is generated then value `none` shall be assumed.]()

<b>Class</b>	<b>QueuedSenderComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes specific to distribution of events (PPortPrototype, SenderReceiverInterface and dataElement carries an "event").			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">PPortComSpec</a> , <a href="#">SenderComSpec</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 4.73: QueuedSenderComSpec**

<b>Class</b>	<b>NonqueuedSenderComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes for non-queued sender/receiver communication (sender side)			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">PPortComSpec</a> , <a href="#">SenderComSpec</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataFilter	<a href="#">DataFilter</a>	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
initValue	<a href="#">ValueSpecification</a>	0..1	aggr	Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.

**Table 4.74: NonqueuedSenderComSpec**

<b>Class</b>	<b>TransmissionComSpecProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	This meta-class defines a set of transmission attributes which the application software is assumed to implement.			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataUpdate Period	<a href="#">TimeValue</a>	0..1	attr	This attribute defines the period in which the application is assumed to transmit the respective data.
minimumSend Interval	<a href="#">TimeValue</a>	0..1	attr	This attribute defines the minimum interval between two consecutive transmissions of the respective data the application is assumed to ensure.
transmission Mode	<a href="#">TransmissionModeDefinitionEnum</a>	0..1	attr	The attribute defines the mode in which the application is assumed to transmit the respective data.

**Table 4.75: TransmissionComSpecProps**

<b>Class</b>	<b>TransmissionAcknowledgementRequest</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Requests transmission acknowledgement that data has been sent successfully. Success/failure is reported via a SendPoint of a RunnableEntity.			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	TransmissionAcknowledgementRequest			
timeout	TimeValue	0..1	attr	Number of seconds before an error is reported or in case of allowed redundancy, the value is sent again.

**Table 4.76: TransmissionAcknowledgementRequest**

**[constr\_1892] Existence of attribute `TransmissionAcknowledgementRequest.timeout`** [For each `TransmissionAcknowledgementRequest`, attribute `timeout` shall exist **at the time when the contract phase generation is executed.**]()

Enumeration	HandleOutOfRangeEnum
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication
<b>Note</b>	A value of this type is taken for controlling the range checking behavior of the AUTOSAR RTE.
<b>Literal</b>	<b>Description</b>
default	The RTE will use the <code>initValue</code> if the actual value is out of the specified bounds. <b>Tags:</b> atp.EnumerationLiteralIndex=0
external Replacement	This indicates that the value replacement is sourced from the attribute <code>replaceWith</code> . <b>Tags:</b> atp.EnumerationLiteralIndex=1
ignore	The RTE will ignore any attempt to send or receive the corresponding <code>dataElement</code> if the value is out of the specified range. <b>Tags:</b> atp.EnumerationLiteralIndex=2
invalid	The RTE will use the <code>invalidValue</code> if the value is out of the specified bounds. <b>Tags:</b> atp.EnumerationLiteralIndex=3
none	A range check is not required. <b>Tags:</b> atp.EnumerationLiteralIndex=4
saturate	The RTE will saturate the value of the <code>dataElement</code> 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. <b>Tags:</b> atp.EnumerationLiteralIndex=5

**Table 4.77: HandleOutOfRangeEnum**

Enumeration	TransmissionModeDefinitionEnum
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication
<b>Note</b>	This meta-class defines possible settings for the transmission mode.
<b>Literal</b>	<b>Description</b>
cyclic	The data is assumed to be transmitted in a cyclic manner. The cycle is defined by <code>dataUpdatePeriod</code> . <b>Tags:</b> atp.EnumerationLiteralIndex=0
cyclicAndOnChange	The data is assumed to be transmitted in a cyclic manner (with cycle time <code>dataUpdatePeriod</code> ) and additionally there may be arbitrary transmission if the data value changes (minimum <code>sendInterval</code> to be respected, if defined). <b>Tags:</b> atp.EnumerationLiteralIndex=2
triggered	The data is assumed to be transmitted in an arbitrary manner (minimum <code>sendInterval</code> to be respected, if defined). <b>Tags:</b> atp.EnumerationLiteralIndex=1

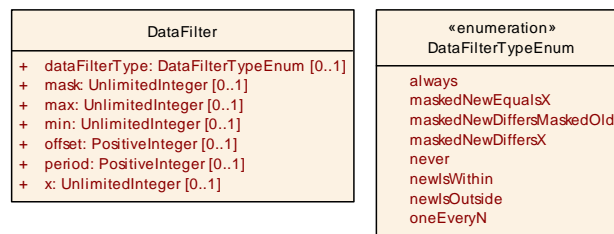
**Table 4.78: TransmissionModeDefinitionEnum**

<b>Class</b>	<b>CompositeNetworkRepresentation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	This meta-class is used to define the network representation of leaf elements of composite application data types.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
leafElement	<a href="#">ApplicationCompositeElementDataPrototype</a>	0..1	iref	This represents that leaf element of an application composite data type.  <b>InstanceRef implemented by:</b> <a href="#">ApplicationCompositeElementInPortInterfaceInstanceRef</a>
networkRepresentation	<a href="#">SwDataDefProps</a>	0..1	aggr	The SwDataDefProps owned by the CompositeNetworkRepresentation are used to define the network representation of the leaf element of an Application CompositeDataType.

**Table 4.79: CompositeNetworkRepresentation**

### 4.5.1.3 Data Filter

Figure 4.37 shows the model of the communication attributes relevant for defining data filters.



**Figure 4.37: DataFilter and its communication attributes.**

**[TPS\_SWCT\_01221] DataFilter** [For every [RPortPrototype](#) or [PPortPrototype](#) typed by a [SenderReceiverInterface](#), a [DataFilter](#) can be defined given that non-queued communication is foreseen.]()

Several filter algorithms formally described by the enumeration type [DataFilterTypeEnum](#) in the meta-model are taken from the ISO 17356-4 specification [17] that is referenced by the RTE specification [2].

**[TPS\_SWCT\_01222] Applicability of DataFilter** [This ISO 17356-4 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. ISO 17356-4 specification [17], [DataFilters](#) can only be applied to values with an integer base type **at any time in the workflow.**]()

<b>Class</b>	<b>DataFilter</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Filter			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataFilterType	DataFilterTypeEnum	0..1	attr	This attribute specifies the type of the filter.
mask	UnlimitedInteger	0..1	attr	Mask for old and new value.
max	UnlimitedInteger	0..1	attr	Value to specify the upper boundary
min	UnlimitedInteger	0..1	attr	Value to specify the lower boundary
offset	PositiveInteger	0..1	attr	Specifies the initial number of messages to occur before the first message is passed
period	PositiveInteger	0..1	attr	Specifies number of messages to occur before the message is passed again
x	UnlimitedInteger	0..1	attr	Value to compare with

**Table 4.80: DataFilter**

**[constr\_1890] Existence of attribute DataFilter.dataFilterType** [For each DataFilter, attribute dataFilterType shall exist at the time when the RTE is generated.]()

<b>Enumeration</b>	<b>DataFilterTypeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Filter
<b>Note</b>	This enum specifies the supported DataFilterTypes.
<b>Literal</b>	<b>Description</b>
always	No filtering is performed so that the message always passes. <b>Tags:</b> atp.EnumerationLiteralIndex=0
maskedNewDiffers 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) <b>Tags:</b> atp.EnumerationLiteralIndex=1
maskedNewDiffers X	Pass messages whose masked value is not equal to a specific value x (new_value&mask) != x new_value: current value of the message <b>Tags:</b> atp.EnumerationLiteralIndex=2
maskedNewEquals X	Pass messages whose masked value is equal to a specific value x (new_value&mask) == x new_value: current value of the message <b>Tags:</b> atp.EnumerationLiteralIndex=3
never	The filter removes all messages. <b>Tags:</b> atp.EnumerationLiteralIndex=4





Enumeration	DataFilterTypeEnum
newIsOutside	Pass a message if its value is outside a predefined boundary. (min > new_value) OR (new_value > max) <b>Tags:</b> atp.EnumerationLiteralIndex=5
newIsWithin	Pass a message if its value is within a predefined boundary. min <= new_value <= max <b>Tags:</b> atp.EnumerationLiteralIndex=6
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). <b>Tags:</b> atp.EnumerationLiteralIndex=7

**Table 4.81: DataFilterTypeEnum**

#### 4.5.1.4 Communication between Application and NV Block Software Components

In many cases, communication between an `NvBlockSwComponentType` (see 11.5.2) and an `ApplicationSwComponentType` is implemented based on mixed communication approaches, i.e. the `ApplicationSwComponentType` most likely uses a `PortPrototype` typed by a `SenderReceiverInterface` while the `NvBlockSwComponentType` uses a `PortPrototype` typed by an `NvDataInterface`.

In such a case it is necessary to clarify the usage of attributes of the respective `ComSpec` on both the sender and receiver side. For this purpose, the constraints `[constr_10071]` and `[constr_10072]` are defined<sup>8</sup>.

**[constr\_10071] Allowed multiplicities of `SenderComSpec` attributes for communication between `ApplicationSwComponentType` and `NvBlockSwComponentType`** [The allowed multiplicities for `SenderComSpec` attributes for a communication between `ApplicationSwComponentType` and `NvBlockSwComponentType` are documented in Table 4.82.

This rule shall be imposed **at the time when the RTE is generated.**]()

<sup>8</sup>As a general rule, queued communication between `ApplicationSwComponentType` and `NvBlockSwComponentType` is not supported by AUTOSAR. The tables 4.82 (referenced by `[constr_10071]`) and 4.83 (referenced by `[constr_10072]`) nevertheless contain a column that represents the case of queued communication for the sake of completeness. But consequently, this column is consistently filled with “n/a” entries in the respective tables.

Sender	ApplicationSwComponentType	
Receiver	NvBlockSwComponentType	
Queuing Configuration	non-queued	queued
SenderComSpec.transmissionAcknowledge	d/c	n/a
SenderComSpec.dataElement	1	n/a
SenderComSpec.handleOutOfRange	d/c	n/a
SenderComSpec.usesEndToEndProtection	d/c	n/a
SenderComSpec.transmissionProps.dataUpdatePeriod	0..1	n/a
SenderComSpec.transmissionProps.minimumSendInterval	0..1	n/a
SenderComSpec.transmissionProps.transmissionMode	0..1	n/a
SenderComSpec.networkRepresentation	d/c	n/a
SenderComSpec.compositeNetworkRepresentation	d/c	n/a
NonqueuedSenderComSpec.dataFilter	d/c	n/a
NonqueuedSenderComSpec.initValue	0..1	n/a

**Table 4.82: Allowed multiplicities of `SenderComSpec` attributes for communication between `ApplicationSwComponentType` and `NvBlockSwComponentType`**

Please note that the abbreviation “d/c” stands for “don’t care”.

**[constr\_10072] Allowed multiplicities of `SenderComSpec` attributes for communication between `NvBlockSwComponentType` and `ApplicationSwComponentType`** [The allowed multiplicities for `SenderComSpec` attributes for a communication between `NvBlockSwComponentType` and `ApplicationSwComponentType` are documented in Table 4.83.

This rule shall be imposed **at the time when the RTE is generated.**]()

Sender	NvBlockSwComponentType	
Receiver	ApplicationSwComponentType	
Queuing Configuration	non-queued	queued
ReceiverComSpec.replaceWith	0	n/a
ReceiverComSpec.dataElement	1	n/a
ReceiverComSpec.receptionProps.dataUpdatePeriod	0	n/a
ReceiverComSpec.receptionProps.timeout	0	n/a
ReceiverComSpec.usesEndToEndProtection	0	n/a
ReceiverComSpec.maxDeltaCounterInit	0	n/a
ReceiverComSpec.handleOutOfRange	0	n/a
ReceiverComSpec.handleOutOfRangeStatus	0	n/a
ReceiverComSpec.maxNoNewOrRepeatedData	0	n/a
ReceiverComSpec.syncCounterInit	0	n/a
ReceiverComSpec.transformationComSpecProps	0	n/a
ReceiverComSpec.networkRepresentation	0	n/a
ReceiverComSpec.compositeNetworkRepresentation	0	n/a
QueuedReceiverComSpec.queueLength	n/a	n/a
NonqueuedReceiverComSpec.filter	0	n/a
NonqueuedReceiverComSpec.timeoutSubstitutionValue	0	n/a







<a href="#">NonqueuedReceiverComSpec.initValue</a>	0..1	n/a
<a href="#">NonqueuedReceiverComSpec.aliveTimeout</a>	0	n/a
<a href="#">NonqueuedReceiverComSpec.enableUpdate</a>	0	n/a
<a href="#">NonqueuedReceiverComSpec.handleDataStatus</a>	0	n/a
<a href="#">NonqueuedReceiverComSpec.handleNeverReceived</a>	0..1	n/a
<a href="#">NonqueuedReceiverComSpec.handleTimeoutType</a>	0	n/a

**Table 4.83: Allowed multiplicities of [ReceiverComSpec](#) attributes for communication between [NvBlockSwComponentType](#) and [ApplicationSwComponentType](#)**

#### 4.5.1.5 Communication behavior to be implemented by the Software Component

AUTOSAR supports the handling of periodic data transmission and reception timeout checking in the Classic AUTOSAR Basic Software layer. The definition of the transmission modes and time values is defined using the [TPS\\_SystemTemplate](#) [10].

The only value which is also available at the [NonqueuedReceiverComSpec](#) is the [aliveTimeout](#) - as this value configures both communication stack timeout handling (in case of remote communication) and RTE timeout handling (in case of local communication).

With the introduction of the transformer technology the aspects of periodic data transmission and checking for timeout on reception may have to be implemented by the application code. The main reason is that the E2E-Transformer needs to be called periodically in order to keep its state machine up to date.

So, the application code needs to call the transmission / reception APIs periodically in order to fulfill these timing requirements. But there may also be other reasons why the application shall take care of the periodicity, for example in case the [LdCom](#) module (which doesn't have any timing features) is used.

The meta-classes [TransmissionComSpecProps](#) and [ReceptionComSpecProps](#) have been introduced to define the expected communication behavior to be implemented by the application code.

As the [TransmissionComSpecProps](#) and [ReceptionComSpecProps](#) define what the expected communication behavior is, the values can also be utilized by communication (network) measurement tools to verify whether the application code actually implements the attributes properly.

The attribute [ReceptionComSpecProps.dataUpdatePeriod](#) defines the time period in which the receiving application shall call the reception API to check for new data.

The attribute `ReceptionComSpecProps.timeout` defines the time after which the application shall assume that the to-be-received data reception has timed out.

The attribute `NonqueuedReceiverComSpec.aliveTimeout` might be defined as well, resulting in duplicate timeout checks.

The communication stack / RTE would check for the value of attribute `aliveTimeout` and the application code would check for the `ReceptionComSpecProps.timeout`. Although it does not seem sensible to perform a timeout check twice, it is not forbidden.

The attribute `TransmissionComSpecProps.dataUpdatePeriod` defines the time period in which the sending application shall call the send API.

The attributes `TransmissionComSpecProps.minimumSendInterval` and `TransmissionComSpecProps.transmissionMode` define values which influence the transmission behavior, implemented by the application code.

Note that the communication stack might also have a periodicity, minimum sending time and transmission mode defined. And such doubled implementation might lead to undesired effects. However AUTOSAR does not regulate the usage of the mechanisms to be exclusive.

## 4.5.2 Communication Specification for Client-Server Communication

### 4.5.2.1 Client ComSpec

The communication aspects relevant for client communication are sketched in Figure 4.38.

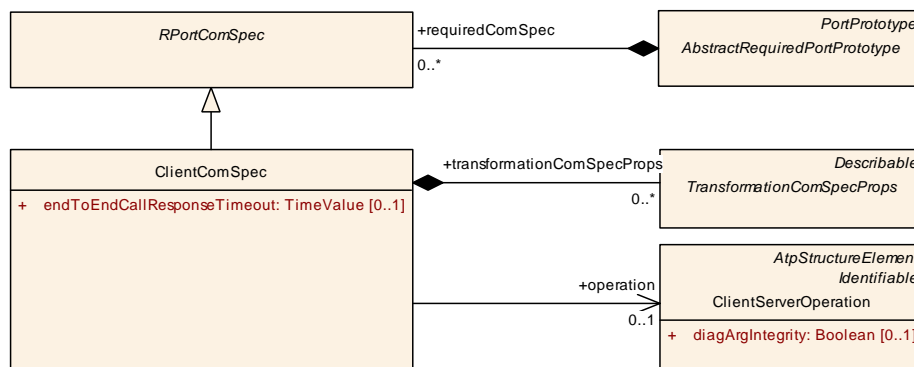


Figure 4.38: Communication attributes of `RPortPrototype` with respect to client-server communication.

**[constr\_1540] Existence of `ClientComSpec.operation`** [The reference `ClientComSpec.operation` shall exist if the `AbstractRequiredPortPrototype` that owns the `ClientComSpec` is typed by a `ClientServerInterface`. This rule shall be imposed **at any time in the workflow.**]()

<b>Class</b>	<b>ClientComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Client-specific communication attributes (RPortPrototype typed by ClientServerInterface).			
<b>Base</b>	ARObject, RPortComSpec			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
endToEndCallResponseTimeout	TimeValue	0..1	attr	This attribute defines the maximum time interval in which the application shall expect the servers's response (time between the sending of the call invocation until the arrival of the server's response).
operation	ClientServerOperation	0..1	ref	This represents the corresponding ClientServerOperation.
transformationComSpecProps	TransformationComSpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

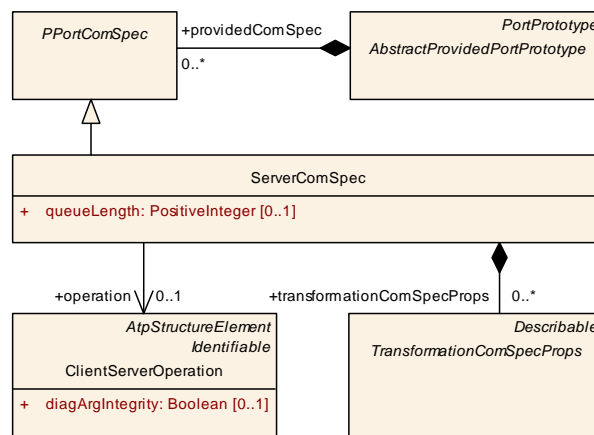
**Table 4.84: ClientComSpec**

Note: on the *AUTOSAR adaptive platform* the `ClientComSpec` can also be used in the context of `RPortPrototypes` typed by `PortInterfaces` that are not available on the *AUTOSAR classic platform*. This is the motivation for the existence of [constr\_1540].

[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*)

#### 4.5.2.2 Server ComSpec

The server side looks very similar but provides an attribute for specifying the queue length.



**Figure 4.39: Communication attributes of `PPortPrototype` with respect to client-server communication.**

<b>Class</b>	<b>ServerComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes for a server port (PPortPrototype and ClientServerInterface).			
<b>Base</b>	ARObject, PPortComSpec			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
operation	ClientServerOperation	0..1	ref	Operation these communication attributes apply to.
queueLength	PositiveInteger	0..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.
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

**Table 4.85: ServerComSpec**

**[constr\_1893] Existence of attribute `ServerComSpec.queueLength`** [For each `ServerComSpec`, attribute `queueLength` shall exist **at the time when the RTE is generated.**]()

**[constr\_1541] Existence of `ServerComSpec.operation`** [The reference `ServerComSpec.operation` shall exist if the `AbstractProvidedPortPrototype` that owns the `ServerComSpec` is typed by a `ClientServerInterface`.

This rule shall be imposed **at any time in the workflow.**]()

Note: on the *AUTOSAR adaptive platform* the `ServerComSpec` can also be used in the context of `RPortPrototypes` typed by `PortInterfaces` that are not available on the *AUTOSAR classic platform*. This is the motivation for the existence of **[constr\_1541]**.

**[TPS\_SWCT\_01225] `RunnableEntity` implements the functionality of more than one `ClientServerOperations`** [A single `RunnableEntity` can implement the functionality of more than one `ClientServerOperations`.

For this purpose, one `OperationInvokedEvent` for each affected `ClientServerOperation` shall reference the respective `RunnableEntity`.

The attribute `ServerComSpec.queueLength` shall be taken for the determination of the resulting queue length, **[constr\_1128]** applies.]()

For more information about requirements towards `ClientServerOperations` triggering the same `RunnableEntity` can be found in section 7.2.4.3, specifically **[constr\_2000]**.

**[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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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*)

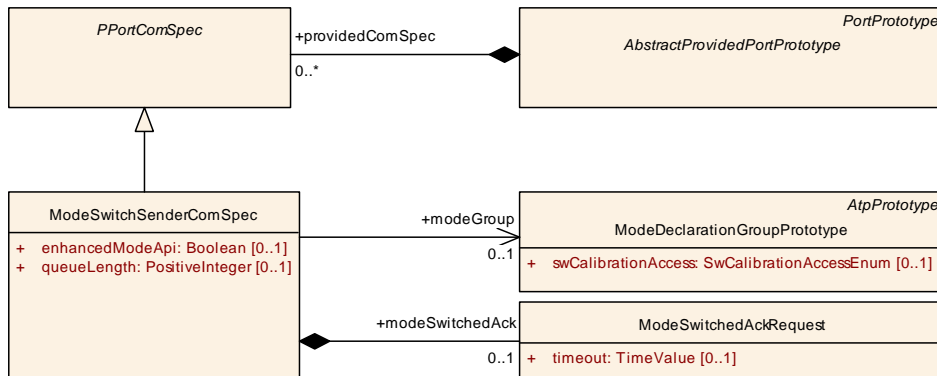
See chapter 4.5.6 for details.

### 4.5.3 Communication Specification for Mode Switch Communication

#### 4.5.3.1 Mode Switch Sender ComSpec

In analogy to the previous section, Figure 4.40 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 4.40: Communication attributes of `PPortPrototype` with respect to mode switch communication.**

<b>Class</b>	<b>ModeSwitchSenderComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes of PPortPrototypes with respect to mode communication			
<b>Base</b>	ARObject, PPortComSpec			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
enhancedMode Api	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	ModeDeclarationGroup Prototype	0..1	ref	ModeDeclarationGroupPrototype (of the same Port Interface) to which these communication attributes apply.





Class	ModeSwitchSenderComSpec			
modeSwitchedAck	<a href="#">ModeSwitchedAckRequest</a>	0..1	aggr	If this aggregation exists an acknowledgement for the successful processing of the mode switch request is required.
queueLength	PositiveInteger	0..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 4.86: ModeSwitchSenderComSpec**

Class	ModeSwitchedAckRequest			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Requests acknowledgements that a mode switch has been proceeded successfully			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
timeout	<a href="#">TimeValue</a>	0..1	attr	Number of seconds before an error is reported or in case of allowed redundancy, the value is sent again.

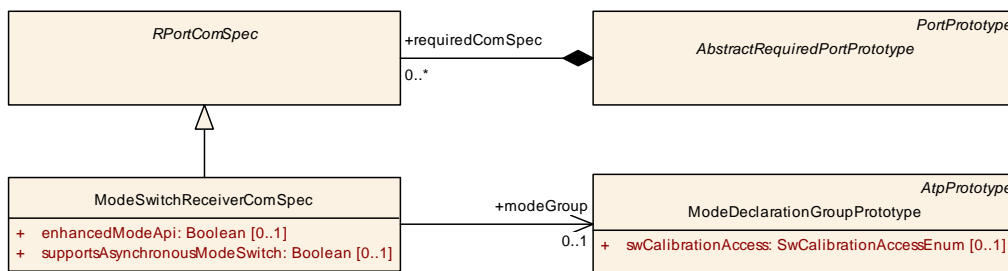
**Table 4.87: ModeSwitchedAckRequest**

**[constr\_1894] Existence of attribute [ModeSwitchSenderComSpec.queueLength](#)**  
 [For each [ModeSwitchSenderComSpec](#), attribute [queueLength](#) shall exist **at the time when the RTE is generated.**]()

**[constr\_1895] Existence of attribute [ModeSwitchSenderComSpec.modeGroup](#)**  
 [For each [ModeSwitchSenderComSpec](#), attribute [modeGroup](#) shall exist **at the time when the contract phase generation is executed.**]()

### 4.5.3.2 Mode Switch Receiver ComSpec

**[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](#))



**Figure 4.41: Communication attributes of [PPortPrototype](#) with respect to mode switch communication.**

<b>Class</b>	<b>ModeSwitchReceiverComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes of RPortPrototypes with respect to mode communication			
<b>Base</b>	ARObject, RPortComSpec			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
enhancedMode Api	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	ModeDeclarationGroup Prototype	0..1	ref	ModeDeclarationGroupPrototype (of the same Port Interface) to which these communication attributes apply.
supports Asynchronous ModeSwitch	Boolean	0..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 4.88: ModeSwitchReceiverComSpec**

**[constr\_1896] Existence of attribute ModeSwitchReceiverComSpec.modeGroup**  
 [For each ModeSwitchReceiverComSpec, attribute modeGroup shall exist at the time when the contract phase generation is executed.]()

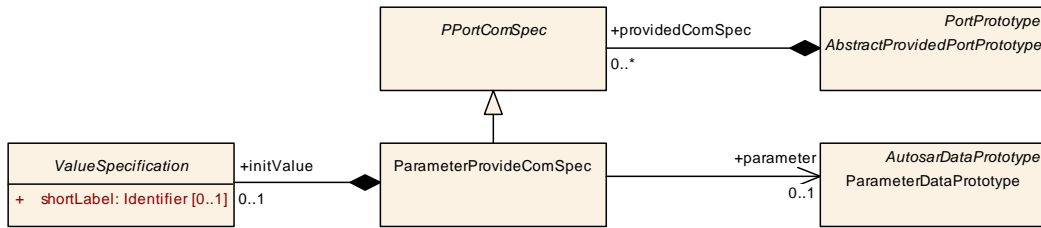
#### 4.5.4 Communication Specification for Parameters

Granted, the definition of a ComSpec for ParameterDataPrototypes 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 ParameterDataPrototypes 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] initialValue 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 initialValue defined in the context of a ParameterProvideComSpec or ParameterRequireComSpec is relevant for connections to the corresponding PortPrototype. An initialValue 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 ParameterDataPrototypes aggregated in a ParameterInterface. The actual modeling is sketched in Figure 4.42 for provided ParameterDataPrototypes and in Figure 4.43 for required ParameterDataPrototypes.

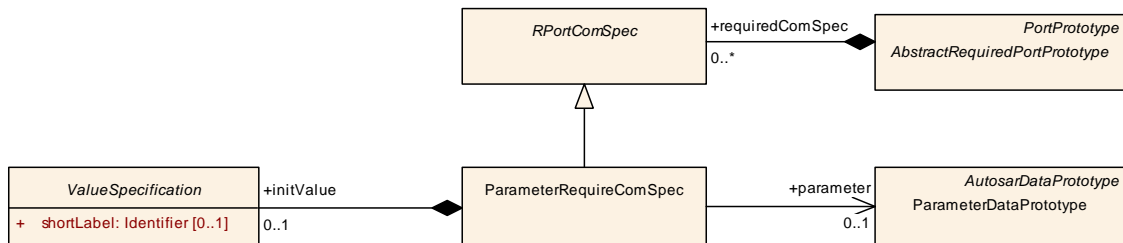


**Figure 4.42: Communication attributes of `ParameterDataPrototypes` with respect to `PPortPrototype`**

<b>Class</b>	<b>ParameterProvideComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	"Communication" specification that applies to parameters on the provided side of a connection.			
<b>Base</b>	<i>ARObject</i> , <i>PPortComSpec</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initValue	<a href="#">ValueSpecification</a>	0..1	aggr	The initial value applicable for the corresponding <code>ParameterDataPrototype</code> .
parameter	<a href="#">ParameterData Prototype</a>	0..1	ref	The <code>ParameterDataPrototype</code> to which the <code>Parameter ComSpec</code> applies.

**Table 4.89: ParameterProvideComSpec**

**[constr\_1897] Existence of reference `ParameterProvideComSpec.parameter`**  
 [For each `ParameterProvideComSpec`, the reference `parameter` shall exist at the time when the contract phase generation is executed.]()



**Figure 4.43: Communication attributes of `ParameterDataPrototypes` with respect to `RPortPrototype`**

<b>Class</b>	<b>ParameterRequireComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	"Communication" specification that applies to parameters on the required side of a connection.			
<b>Base</b>	<i>ARObject</i> , <i>RPortComSpec</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initValue	<a href="#">ValueSpecification</a>	0..1	aggr	The initial value applicable for the corresponding <code>ParameterDataPrototype</code> .
parameter	<a href="#">ParameterData Prototype</a>	0..1	ref	The <code>ParameterDataPrototype</code> to which the <code>Parameter RequireComSpec</code> applies.

**Table 4.90: ParameterRequireComSpec**



[constr\_1898] Existence of reference **ParameterRequireComSpec.parameter** [For each **ParameterRequireComSpec**, the reference **parameter** shall exist at the time when the contract phase generation is executed.]()

## 4.5.5 Communication Specification for NV Data

### 4.5.5.1 NV Require ComSpec

[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)

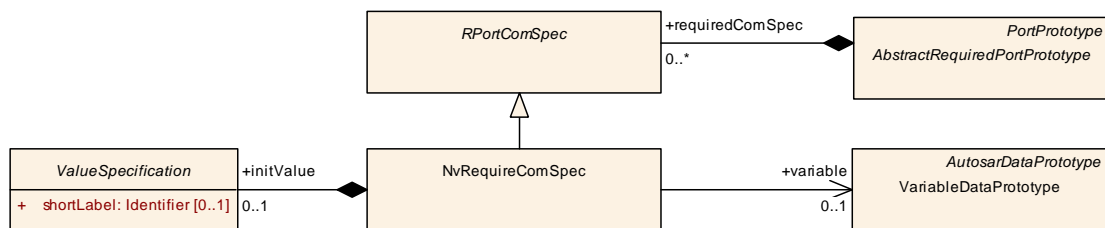


Figure 4.44: Communication attributes of a required **VariableDataPrototypes** used in the context of an **NvDataInterface**

[TPS\_SWCT\_01754] **initValue** defined in the context of a **ComSpec** [Unless [TPS\_SWCT\_01755] applies, 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** shall be ignored anyway.](RS\_SWCT\_03225)

[TPS\_SWCT\_01755] Duplicate existence of **initValue** in the context of a **PR-PortPrototype** typed by an **NvDataInterface** [If an **initValue** is defined in a **NvRequireComSpec** owned by a **PRPortPrototype**, its value shall be ignored.

Instead, the **initValue** shall be taken from the **NvProvideComSpec.ramBlock-InitValue**.](RS\_SWCT\_03225)

<b>Class</b>	<b>NvRequireComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes of RPortPrototypes with respect to Nv data communication on the required side.			
<b>Base</b>	ARObject, RPortComSpec			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	NvRequireComSpec			
initValue	ValueSpecification	0..1	aggr	The initial value owned by the NvComSpec
variable	VariableDataPrototype	0..1	ref	The VariableDataPrototype the ComSpec applies for.

**Table 4.91: NvRequireComSpec**

**[constr\_1899] Existence of reference NvRequireComSpec.variable** [For each NvRequireComSpec, the reference variable shall exist at the time when the contract phase generation is executed.]()

#### 4.5.5.2 NV Provide ComSpec

**[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 is defined.

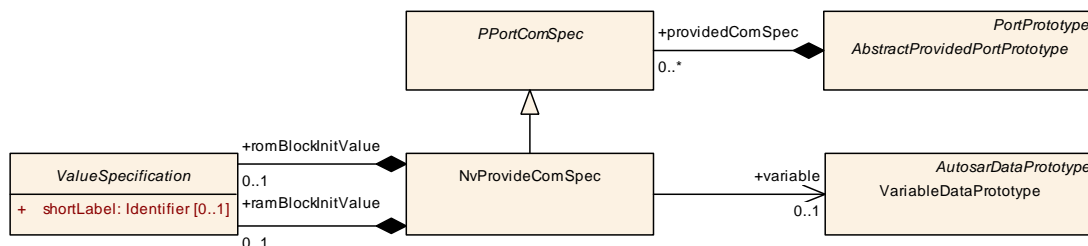
The main purpose of this kind of ComSpec is the definition of initial values for the RAM Block and the ROM Block that corresponds to an nvData defined in the context of the NvDataInterface used to type the given AbstractProvidedPortPrototype.](RS\_SWCT\_03225)

More information about NvProvideComSpec please refer to Figure 4.45.

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 11.9.

Also, note that the romBlockInitValue provided in the NvProvideComSpec does not necessarily have to be identical to the respective section within romBlock in the NvBlockDescriptor.

This could happen if an NvBlockSwComponentType is already existing and an ApplicationSwComponentType is connected to it. Finally, the romBlock inside the NvBlockDescriptor is the only relevant information for the RTE generation.



**Figure 4.45: 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`.

<b>Class</b>	<b>NvProvideComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes of PPortPrototypes with respect to Nv data communication on the provided side.			
<b>Base</b>	<i>ARObject, PPortComSpec</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
ramBlockInit Value	<a href="#">ValueSpecification</a>	0..1	aggr	This represents the initial value of the RAM Block that corresponds to the referenced variable.
romBlockInit Value	<a href="#">ValueSpecification</a>	0..1	aggr	This represents the initial value of the ROM block that corresponds to the referenced variable.
variable	<a href="#">VariableDataPrototype</a>	0..1	ref	This represents the variable for which the ComSpec is specified.

**Table 4.92: NvProvideComSpec**

**[constr\_1900] Existence of reference `NvProvideComSpec.variable`** [For each `NvProvideComSpec`, the reference `variable` shall exist **at the time when the contract phase generation is executed**.]()

#### 4.5.6 Configuration of Data Transformation

The big picture of data transformation in AUTOSAR is explained in the TPS System Template [10]. This chapter focuses on the aspects of data transformation that are related to the configuration of software-components.

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`)

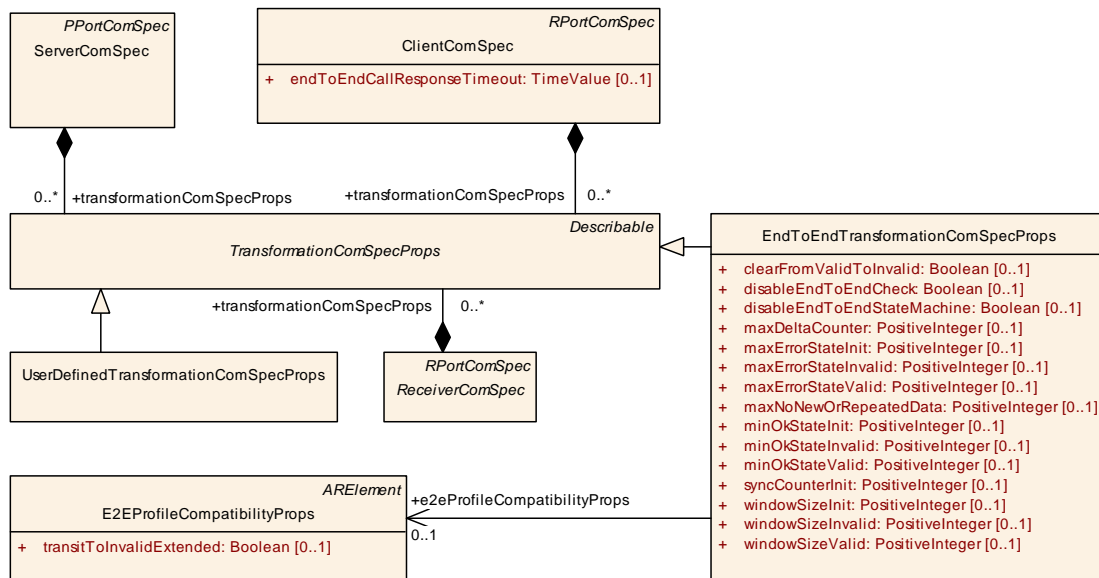
] (*RS\_SWCT\_03221*)

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 4.46: Specification of data transformation properties within ReceiverComSpec, ServerComSpec, and ClientComSpec**

<b>Class</b>	<i>TransformationComSpecProps</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	TransformationComSpecProps holds all the attributes for transformers that are port specific.			
<b>Base</b>	ARObject, Describable			
<b>Subclasses</b>	EndToEndTransformationComSpecProps, UserDefinedTransformationComSpecProps			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 4.93: 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)

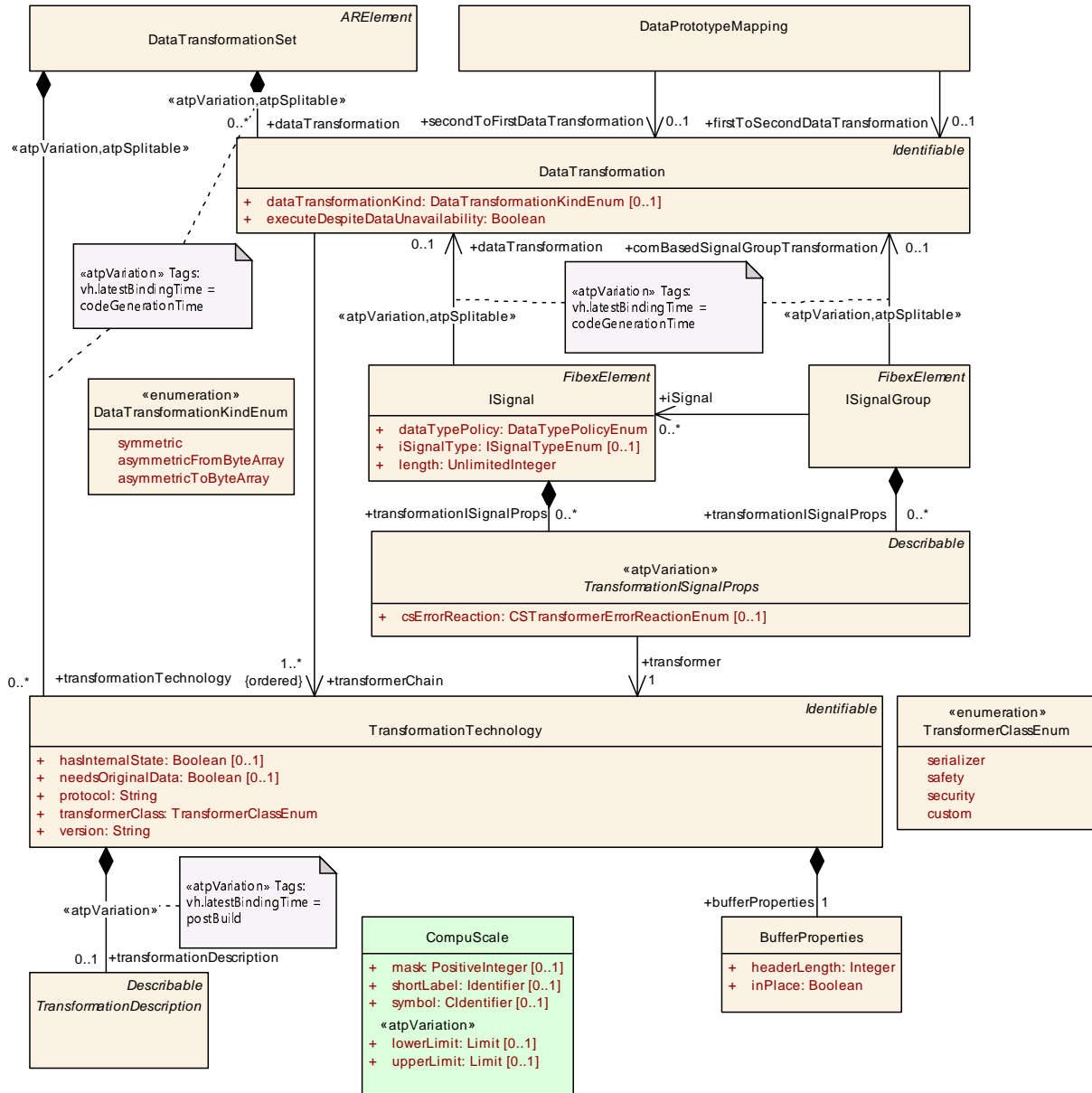


Figure 4.47: Big picture of data transformation in the AUTOSAR meta-model

[constr\_1400] Reference to a specific **DataTransformation** [A specific **DataTransformation** shall only be referenced by either

- a **DataPrototypeMapping** in the role **firstToSecondDataTransformation** (and potentially **secondToFirstDataTransformation**) **or**
- an **ISignal** in the role **dataTransformation** **or**
- an **ISignalGroup** in the role **comBasedSignalGroupTransformation** **or**
- a **ClientServerOperationMapping** in the role **firstToSecondDataTransformation**

This rule shall be imposed **at the time when the RTE is generated.**]()

**[constr\_1401] Restrictions on the relation between DataPrototypeMapping and DataTransformation** [A VariableDataPrototype in the context of a PortPrototype shall –**at the time when the RTE is generated**– 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.

<b>Class</b>	<b>TransformationTechnology</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Transformer			
<b>Note</b>	A TransformationTechnology is a transformer inside a transformer chain. <b>Tags:</b> xml.namePlural=TRANSFORMATION-TECHNOLOGIES			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
bufferProperties	BufferProperties	1	aggr	Aggregation of the mandatory BufferProperties.
hasInternalState	Boolean	0..1	attr	This attribute defines whether the Transformer has an internal state or not.
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 Transformer Description. <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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 4.94: TransformationTechnology**

<b>Class</b>	<b>UserDefinedTransformationComSpecProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	The UserDefinedTransformationComSpecProps is used to specify port specific configuration properties for custom transformers.			
<b>Base</b>	ARObject, Describable, TransformationComSpecProps			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 4.95: UserDefinedTransformationComSpecProps**

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.

**[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	<a href="#">ARObject</a> , <a href="#">Describable</a> , <a href="#">TransformationComSpecProps</a>			
Attribute	Type	Mult.	Kind	Note
clearFromValidToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
disableEndToEndCheck	Boolean	0..1	attr	Disables/Enables the E2E check. The E2Eheader is removed from the payload independent from the setting of this attribute.
disableEndToEndStateMachine	Boolean	0..1	attr	Disables the E2EStateMachine (only E2E check functionality is performed)
e2eProfileCompatibilityProps	<a href="#">E2EProfileCompatibilityProps</a>	0..1	ref	Reference to additional settings for the E2E state machine.
maxDeltaCounter	PositiveInteger	0..1	attr	Maximum allowed difference between two counter values of two consecutively received valid messages. For example, if the receiver gets data with counter 1 and MaxDeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
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.
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. The minimum value is 0.
maxNoNewOrRepeatedData	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.





Class	EndToEndTransformationComSpecProps			
minOkState Invalid	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.
minOkState Valid	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.
windowSizeInit	PositiveInteger	0..1	attr	Size of the monitoring window of state Init for the E2E state machine.
windowSize Invalid	PositiveInteger	0..1	attr	Size of the monitoring window of state Invalid for the E2E state machine.
windowSize Valid	PositiveInteger	0..1	attr	Size of the monitoring window of state Valid for the E2E state machine.

**Table 4.96: 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](#))

[TPS\_SWCT\_01812] **Conditional relevance of attribute [EndToEndTransformationComSpecProps.disableEndToEndStateMachine](#)** [If [EndToEndTransformationComSpecProps.disableEndToEndCheck](#) is set to True then the value [EndToEndTransformationComSpecProps.disableEndToEndStateMachine](#) shall be ignored.] ([RS\\_SWCT\\_03221](#))

Class	E2EProfileCompatibilityProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class collects settings for configuration of the E2E state machine. <b>Tags:</b> atp.recommendedPackage=E2EProfileCompatibilityPropsCollection			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
transitToInvalid Extended	Boolean	0..1	attr	E2E State machine behavior concerning transition from NODATA/INIT to INVALID  value=0 (false): no direct transition from NODATA to INVALID, no transition from INIT to INVALID due to counter-related faults (Autosar R19-11 or former behavior)  value=1 (true): direct transition from NODATA to INVALID covered, transition from INIT to INVALID due to counter-related faults covered (state machine extended)

**Table 4.97: E2EProfileCompatibilityProps**



[constr\_5234] Existence of attribute **E2EProfileCompatibilityProps.transitToInvalidExtended** is mandatory for each **EndToEndTransformationComSpecProps** [For each **EndToEndTransformationComSpecProps**, a reference in the role **e2eProfileCompatibilityProps** to meta-class **E2EProfileCompatibilityProps** shall exist and the referenced **E2EProfileCompatibilityProps** shall define a value for the attribute **transitToInvalidExtended** at the time when the RTE is generated.]()

### 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**)

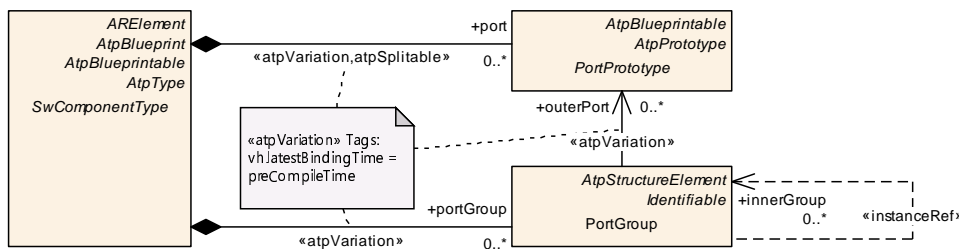


Figure 4.48: Declaration of **PortGroups**

<b>Class</b>	<b>PortGroup</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	<p>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.</p> <p>A <b>PortGroup</b> 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 <b>PortGroup</b> within an atomic SWC cannot be linked to inner groups.</p>			
<b>Base</b>	<i>ARObject</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
innerGroup	<a href="#">PortGroup</a>	*	iref	Links a <b>PortGroup</b> in a composition to another <b>PortGroup</b> , that is defined in a component which is part of this <b>CompositionSwComponentType</b> .  <b>InstanceRef implemented by:</b> <a href="#">InnerPortGroupInCompositionInstanceRef</a>





Class	PortGroup			
outerPort	PortPrototype	*	ref	Outer PortPrototype of this AtomicSwComponentType which belongs to the group. A port can belong to several groups or to no group at all.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 4.98: 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.] ([RS\\_SWCT\\_03200](#), [RS\\_SWCT\\_03201](#))

For more details, see chapter [7.11.2](#).

**[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

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

## 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 [10] where the details of the alternative approach are explained.

Caveat: The E2E wrapper approach involves technologies that are not subjected to the AUTOSAR standard and is superseded by the superior E2E transformer approach (which is fully standardized by AUTOSAR, see [18]). Hence, new projects (without legacy constraints due to carry-over parts) shall use the fully standardized E2E transformer approach.

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](#))

<b>Class</b>	<b>EndToEndDescription</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::EndToEndProtection			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
category	NameToken	0..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. <b>Tags:</b> xml.sequenceOffset=-100





Class	EndToEndDescription			
counterOffset	PositiveInteger	0..1	attr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=-50</p>
crcOffset	PositiveInteger	0..1	attr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=-60</p>
dataId (ordered)	PositiveInteger	*	attr	<p>This represents a unique numerical identifier.</p> <p>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 EndToEnd Protection.</p> <p><b>Tags:</b>xml.sequenceOffset=-90</p>
dataIdMode	PositiveInteger	0..1	attr	<p>There are three inclusion modes how the implicit two-byte Data ID is included in the one-byte CRC:</p> <ul style="list-style-type: none"> <li>• dataIdMode = 0: Two bytes are included in the CRC (double ID configuration) This is used in variant 1A.</li> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> </ul> <p><b>Tags:</b>xml.sequenceOffset=-85</p>
dataIdNibble Offset	PositiveInteger	0..1	attr	<p>Bit offset of the low nibble of the high byte of Data ID. The applicability of this attribute is controlled by <a href="#">[constr_1261]</a>.</p> <p><b>Tags:</b>xml.sequenceOffset=-25</p>
dataLength	PositiveInteger	0..1	attr	<p>This attribute represents the length of the Array representation of the Signal Group/VariableDataPrototype including CRC and Counter in bits.</p> <p><b>Tags:</b>xml.sequenceOffset=-80</p>





Class	EndToEndDescription			
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.  <b>Tags:</b> xml.sequenceOffset=-70
maxNoNewOrRepeatedData	PositiveInteger	0..1	attr	The maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions.  <b>Tags:</b> 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.  <b>Tags:</b> xml.sequenceOffset=-30

**Table 4.99: EndToEndDescription**

**[constr\_1901] Existence of attribute `EndToEndDescription.category`** [For each `EndToEndDescription`, attribute `category` shall exist **at the time when the contract phase generation is executed.**]()

**[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 [10]

] (`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).
- **[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](#))

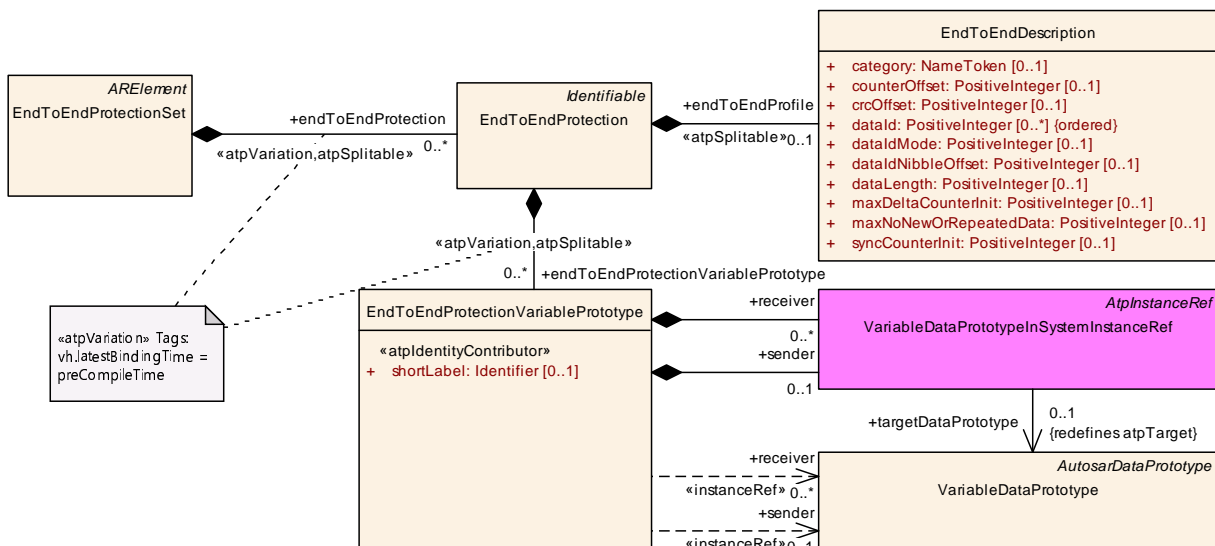
**[TPS\_SWCT\_01092] EndToEndProtectionSet** [The meta-class `EndToEndProtectionSet` provides a container for `EndToEndProtection`. The aggregation is stereotyped `<<atpSplittable>>` because the information about end-to-end protection is added at a later step in the development workflow.] ([RS\\_SWCT\\_03240](#))

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 splittable** [`EndToEndProtection` aggregates `EndToEndDescription` using stereotype `<<atpSplittable>>`].

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 4.49: 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`

**at the time when the contract phase generation is executed.]()**

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].

This rule shall be imposed at the time when the contract phase generation is executed.]()

**[constr\_1112] Constraints of `dataIdMode` in `PROFILE_01`** [In `PROFILE_01`, the applicable range of values for `dataIdMode` is [0 .. 3].

This rule shall be imposed at the time when the contract phase generation is executed.]()

**[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  $value \bmod 4 = 0$  applies.

This rule shall be imposed at the time when the contract phase generation is executed.]()

**[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  $value \bmod 4 = 0$  applies.

This rule shall be imposed at the time when the contract phase generation is executed.]()

**[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  $value \bmod 8 = 0$  applies.

This rule shall be imposed at the time when the contract phase generation is executed.]()



**[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].

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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].

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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].

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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 `RPortPrototype.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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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`

**at the time when the contract phase generation is executed**]()

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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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].

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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].

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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].

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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].

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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.]()

<b>Class</b>	<b>EndToEndProtectionSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::EndToEndProtection			
<b>Note</b>	This represents a container for collection EndToEndProtectionInformation. <b>Tags:</b> atp.recommendedPackage=EndToEndProtectionSets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
endToEndProtection	<a href="#">EndToEndProtection</a>	*	aggr	This is one particular EndToEndProtection. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=endToEndProtection.shortName, endToEndProtection.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

**Table 4.100: EndToEndProtectionSet**

<b>Class</b>	<b>EndToEndProtection</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::EndToEndProtection			
<b>Note</b>	This meta-class represents the ability to describe a particular end to end protection.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
endToEndProfile	<a href="#">EndToEndDescription</a>	0..1	aggr	This represents the particular EndToEndDescription. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=endToEndProfile
endToEndProtectionISignalPdu	<a href="#">EndToEndProtectionISignalPdu</a>	*	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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=endToEndProtectionISignalPdu, endToEndProtectionISignalPdu.variationPoint.shortLabel vh.latestBindingTime=preCompileTime





Class	EndToEndProtection			
endToEndProtectionVariablePrototype	<a href="#">EndToEndProtectionVariablePrototype</a>	*	aggr	<p>Defines to which VariableDataPrototypes in the roles of one sender and one or more receivers this EndTo Endprotection applies.</p> <p>It shall be possible to aggregate several EndToEnd ProtectionVariablePrototype 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.</p> <p>Caveat: The E2E wrapper approach involves technologies that are not subjected to the AUTOSAR standard and is superseded by the superior E2E transformer approach (which is fully standardized by AUTOSAR). Hence, new projects (without legacy constraints due to carry-over parts) shall use the fully standardized E2E transformer approach.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=endToEndProtectionVariablePrototype.shortLabel, endToEndProtectionVariablePrototype.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>

**Table 4.101: EndToEndProtection**

[constr\_1902] Existence of attribute [EndToEndProtection.endToEndProfile](#) [For each [EndToEndProtection](#), attribute [endToEndProfile](#) shall exist at the time when the contract phase generation is executed.]()

Class	EndToEndProtectionVariablePrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::EndToEndProtection			
Note	<p>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.</p> <p>Caveat: The E2E wrapper approach involves technologies that are not subjected to the AUTOSAR standard and is superseded by the superior E2E transformer approach (which is fully standardized by AUTOSAR). Hence, new projects (without legacy constraints due to carry-over parts) shall use the fully standardized E2E transformer approach.</p>			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
receiver	<a href="#">VariableDataPrototype</a>	*	iref	<p>This represents the receiver. Note that 1:n communication is supported for this use case.</p> <p><b>InstanceRef implemented by:</b>VariableDataPrototypeInSystemInstanceRef</p>
sender	<a href="#">VariableDataPrototype</a>	0..1	iref	<p>This represents the sender.</p> <p>Can be optional if an ecu extract is provided and the sender is part of the extract.</p> <p><b>InstanceRef implemented by:</b>VariableDataPrototypeInSystemInstanceRef</p>





Class		EndToEndProtectionVariablePrototype		
shortLabel	Identifier	0..1	attr	This serves as part of the split key in case of more than one EndToEndProtectionVariablePrototype is aggregated in the bound model. <b>Stereotypes:</b> atpIdentityContributor

**Table 4.102: 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 at the time when the contract phase generation is executed.]()

## 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 are used to specify the grouping of PortPrototypes to the higher conceptual level of a Virtual Function Cluster.] (RS\_SWCT\_03241, RS\_SWCT\_03201)

Please note that more information regarding the semantics of PortGroups can be found in chapter 4.6.

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 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`, see [constr\_1147].

#### 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` [21]

](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 **and** refers to a `PortGroup` (that comprises the VFC) in the role `representedPortGroup`.](RS\_SWCT\_03241, RS\_SWCT\_03201)

For further information, please refer to [TPS\_SWCT\_01126].

#### 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` [21]

]([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-component<sup>9</sup>.

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.

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 **and** refers to a [PortGroup](#) (that comprises the VFC) in the role `representedPortGroup`.]([RS\\_SWCT\\_03241](#), [RS\\_SWCT\\_03201](#))

For further information, please refer to [[TPS\\_SWCT\\_01126](#)].

## 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 condense to two basic aspects:

---

<sup>9</sup>The 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](#).



- **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`.

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 `dpgDoesNotRequireCoherency` 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 `dpgDoesNotRequireCoherency` 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.]([RS\\_SWCT\\_03065](#))

[TPS\_SWCT\_01479] **Applicability of ConsistencyNeeds** [ConsistencyNeeds can only be applied to RunnableEntities that make use of “implicit” communication.] (RS\_SWCT\_03065)

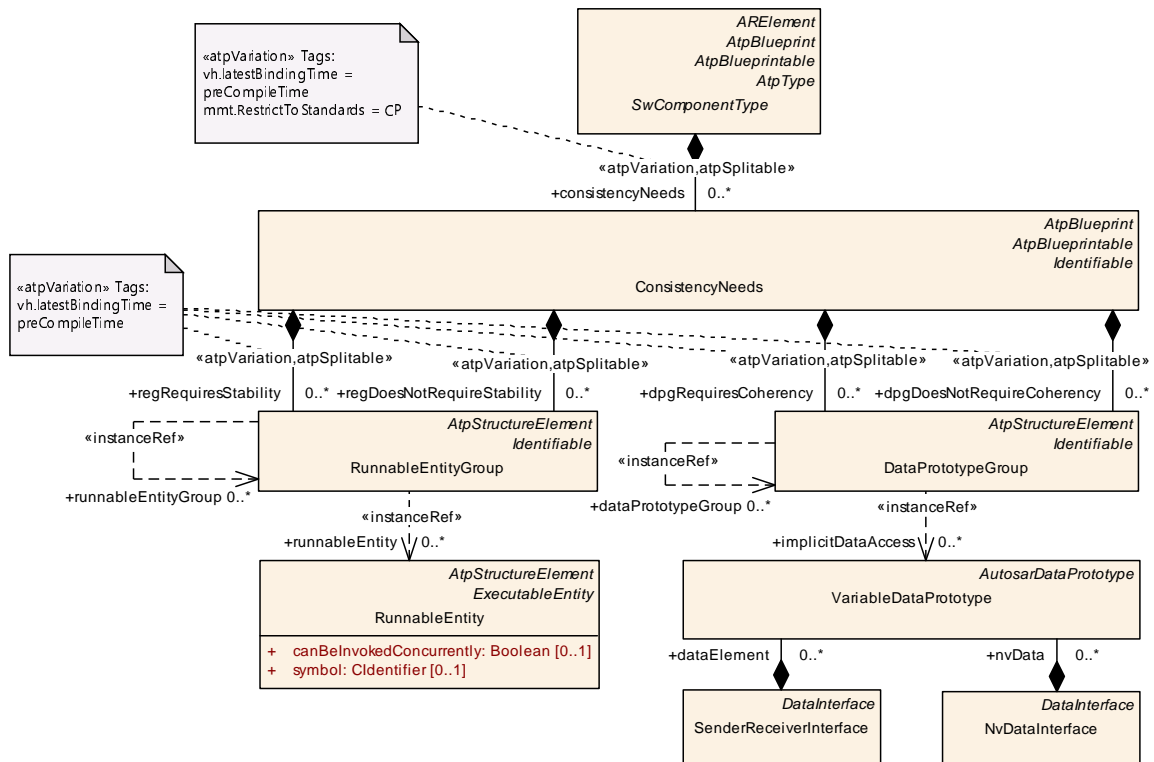


Figure 4.50: Formal definition of implicit communication behavior

[TPS\_SWCT\_01466] **ConsistencyNeeds applied on RunnableEntities that do not use implicit communication** [If a ConsistencyNeeds is applied on RunnableEntities 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 RunnableEntity 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*.

This rule shall be imposed **at the time when the creation of the *CompositionSwComponentType* is finished.**]()

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* **at the time when the contract phase generation is executed.**]()

Strictly speaking, these are the *RunnableEntities* 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].

<b>Class</b>	<b>ConsistencyNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
<b>Note</b>	This meta-class represents the ability to define requirements on the implicit communication behavior.			
<b>Base</b>	<i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dpgDoesNotRequireCoherency	<a href="#">DataPrototypeGroup</a>	*	aggr	This group of <i>VariableDataPrototypes</i> does not require coherency with respect to the implicit communication behavior.  <b>Stereotypes:</b> <i>atpSplitable</i> ; <i>atpVariation</i> <b>Tags:</b> <i>atp.Splitkey=dpgDoesNotRequireCoherency.shortName</i> , <i>dpgDoesNotRequireCoherency.variationPoint.shortLabel</i> <i>vh.latestBindingTime=preCompileTime</i>
dpgRequiresCoherency	<a href="#">DataPrototypeGroup</a>	*	aggr	This group of <i>VariableDataPrototypes</i> requires coherency with respect to the implicit communication behavior, i.e. all read and write access to <i>VariableDataPrototypes</i> in the <i>DataPrototypeGroup</i> by the <i>RunnableEntities</i> of the <i>RunnableEntityGroup</i> need to be handled in a coherent manner.  <b>Stereotypes:</b> <i>atpSplitable</i> ; <i>atpVariation</i> <b>Tags:</b> <i>atp.Splitkey=dpgRequiresCoherency.shortName</i> , <i>dpgRequiresCoherency.variationPoint.shortLabel</i> <i>vh.latestBindingTime=preCompileTime</i>





Class	ConsistencyNeeds			
regDoesNotRequireStability	<a href="#">RunnableEntityGroup</a>	*	aggr	<p>This group of RunnableEntities does not require stability with respect to the implicit communication behavior.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=regDoesNotRequireStability.shortName, regDoesNotRequireStability.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
regRequiresStability	<a href="#">RunnableEntityGroup</a>	*	aggr	<p>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 RunnableEntities of the RunnableEntityGroup need to be handled in a stable manner.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=regRequiresStability.shortName, regRequiresStability.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>

**Table 4.103: ConsistencyNeeds**

Class	RunnableEntityGroup			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
<b>Note</b>	This meta-class represents the ability to define a collection of RunnableEntities. The collection can be nested.			
<b>Base</b>	<i>ARObject</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
runnableEntity	<a href="#">RunnableEntity</a>	*	iref	<p>This represents a collection of RunnableEntities that belong to the enclosing RunnableEntityGroup.</p> <p><b>Stereotypes:</b> atpVariation</p> <p><b>Tags:</b>vh.latestBindingTime=preCompileTime</p> <p><b>InstanceRef implemented by:</b><a href="#">RunnableEntityInCompositionInstanceRef</a></p>
runnableEntityGroup	<a href="#">RunnableEntityGroup</a>	*	iref	<p>This represents the ability to define nested groups of RunnableEntities.</p> <p><b>Stereotypes:</b> atpVariation</p> <p><b>Tags:</b>vh.latestBindingTime=preCompileTime</p> <p><b>InstanceRef implemented by:</b><a href="#">InnerRunnableEntityGroupInCompositionInstanceRef</a></p>

**Table 4.104: RunnableEntityGroup**

Class	DataPrototypeGroup			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note





Class	DataPrototypeGroup			
dataPrototype Group	<a href="#">DataPrototypeGroup</a>	*	iref	This represents the ability to define nested groups of VariableDataPrototypes. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime <b>InstanceRef implemented by:</b> <a href="#">InnerDataPrototypeGroup</a> <a href="#">InCompositionInstanceRef</a>
implicitData Access	<a href="#">VariableDataPrototype</a>	*	iref	This represents a collection of VariableDataPrototypes that belong to the enclosing DataPrototypeGroup <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime <b>InstanceRef implemented by:</b> <a href="#">VariableDataPrototypeInCompositionInstanceRef</a>

**Table 4.105: DataPrototypeGroup**

#### 4.9.1 Consistency Needs on Receiver Side

**[TPS\_SWCT\_01472] Receiving [SwComponentType](#) owns a [DataPrototypeGroup](#) in the role [dpgRequiresCoherency](#)** [If a receiving [SwComponentType](#) owns a [DataPrototypeGroup](#) in the role [dpgRequiresCoherency](#) for one or several of its [RunnableEntity](#)s 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 [RunnableEntity](#)s it is required that the values of implicitly communicated [VariableDataPrototypes](#) are kept stable over the execution of all [RunnableEntity](#)s 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 [dpgRequiresCoherency](#)** [If a receiving [SwComponentType](#) owns a [RunnableEntityGroup](#) in the role [regRequiresStability](#) and also owns one or several [DataPrototypeGroups](#) in the role [dpgRequiresCoherency](#) 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 [RunnableEntity](#)s belonging to the given [RunnableEntityGroup](#).] ([RS\\_SWCT\\_03065](#))

#### 4.9.2 Consistency Needs on Sender Side

**[TPS\_SWCT\_01475] Sending `SwComponentType` owns a `DataPrototypeGroup` in the role `dpgRequiresCoherency`** [If a sending `SwComponentType` owns a `DataPrototypeGroup` in the role `dpgRequiresCoherency` 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 do not belong 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 `dpgRequiresCoherency` and also `RunnableEntityGroups`** [If a sending `SwComponentType` owns a `DataPrototypeGroup` in the role `dpgRequiresCoherency`, `RunnableEntityGroups` in the role `regDoesNotRequireStability` may exist.

Read accesses from `RunnableEntities` in those `RunnableEntityGroups` will not read the coherent values but the intermediary values written by `RunnableEntities` 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](#))

<b>Application Data Level</b>
<b>Implementation Data Level</b>
<b>Base Type Level</b>

**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 focus<sup>1</sup> 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](#))

Please note that [ApplicationDataTypes](#) – by virtue of being platform-independent by definition – do not become visible as data types in the code implementation of software-components.

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

<sup>1</sup>There are some aspects that affect the RTE, e.g. scaling of [dataElements](#)

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.

- 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. 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](#))

Further information about the compatibility requirements between application level and implementation level can be found in section [6.2.5](#).

**[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, the *Implementation Data* level can be used in the description of interfaces, and data (e.g. debug data) within the basic software, see [6] 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 [22] is responsible for this.

It implies that the configuration depends on the definition of the data that are transmitted between components<sup>2</sup>.] ([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 is 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](#).

---

<sup>2</sup>More exactly speaking, the data shall be converted to and from a so-called [SystemSignal](#).

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.

|(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 [5.6](#).
- Properties for [ApplicationDataTypes](#) are summarized in table [5.7](#).
- Properties for [ImplementationDataTypes](#) are summarized in table [5.17](#).
- Properties for [DataPrototypes](#) typed by [ApplicationDataTypes](#) are summarized in table [5.34](#).
- Properties for [DataPrototypes](#) typed by [ImplementationDataTypes](#) are summarized in table [5.35](#).
- Applicability of [SwDataDefProps](#) is summarized in table [5.43](#).

## 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 [Autosar-DataType](#), from which an [ApplicationDataType](#) and an [Implementation-DataType](#) is derived.](RS\_SWCT\_03215, RS\_SWCT\_03216, RS\_SWCT\_03217)

[TPS\_SWCT\_01073] **Composite ApplicationDataType** [An [Application-DataType](#) 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)

[TPS\_SWCT\_01074] Composite **ImplementationDataType** [An **ImplementationDataType** can also be composed of elements but in this case no type/prototype concept (see [11]) has been applied. Both concepts will be explained in the following chapters in more detail.] ([RS\\_SWCT\\_03215](#), [RS\\_SWCT\\_03217](#))

Figure 5.1 shows a summary of the basic meta-classes used for the definition of **AutosarDataTypes**.

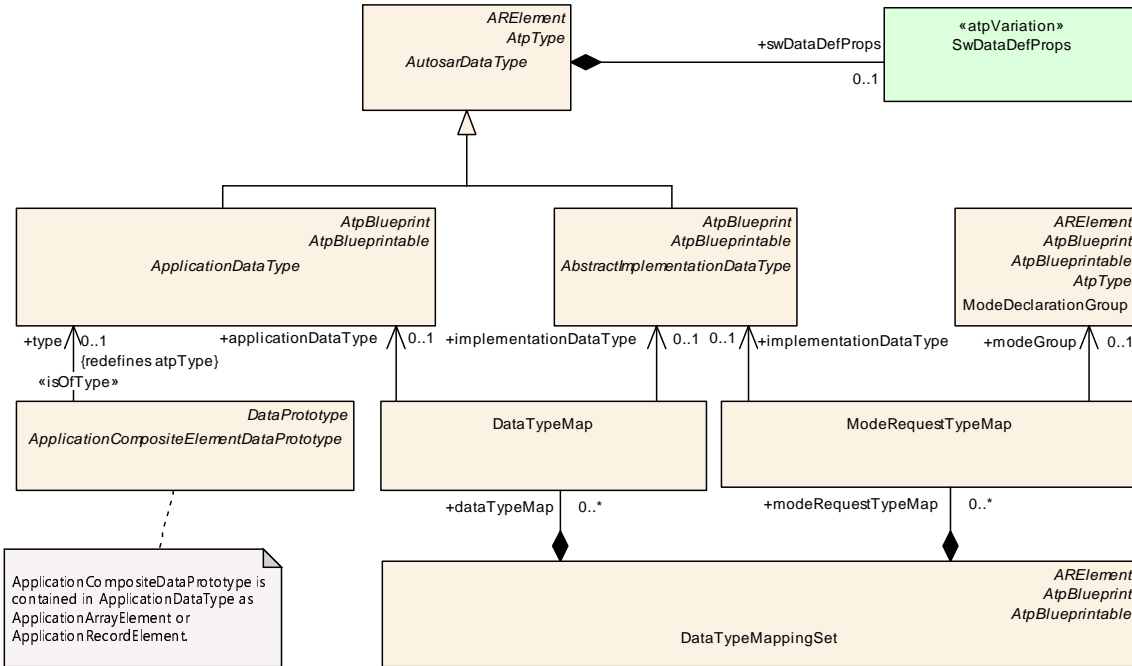


Figure 5.1: Summary of **AutosarDataType**

<b>Class</b>	<b>AutosarDataType</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	Abstract base class for user defined AUTOSAR data types for software.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> , <a href="#">Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">AbstractImplementationDataType</a> , <a href="#">ApplicationDataType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swDataDef Props	<a href="#">SwDataDefProps</a>	0..1	aggr	The properties of this AutosarDataType.

Table 5.2: **AutosarDataType**

<b>Class</b>	<b>ApplicationDataType</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			





<b>Class</b>	<b>ApplicationDataType</b> (abstract)			
<b>Note</b>	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.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">ApplicationCompositeDataType</a> , <a href="#">ApplicationPrimitiveDataType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 5.3: ApplicationDataType**

## 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 other in order to describe both aspects of one [dataElement](#).] ([RS\\_SWCT\\_03216](#), [RS\\_SWCT\\_03217](#), [RS\\_SWCT\\_03215](#))

<b>Class</b>	<b>DataTypeMap</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	This class represents the relationship between <a href="#">ApplicationDataType</a> and its implementing <a href="#">AbstractImplementationDataType</a> .			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
applicationData Type	<a href="#">ApplicationDataType</a>	0..1	ref	This is the corresponding <a href="#">ApplicationDataType</a>
implementation DataType	<a href="#">AbstractImplementation DataType</a>	0..1	ref	This is the corresponding <a href="#">AbstractImplementationData Type</a> .

**Table 5.4: 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.

[constr\_1903] **Existence of reference [DataTypeMap.applicationDataType](#)** [For each [DataTypeMap](#), reference [applicationDataType](#) shall exist **at the time when the contract phase generation is executed.**] ()

**[constr\_1904] Existence of reference `DataTypeMap.implementationDataType`**  
 [For each `DataTypeMap`, reference `implementationDataType` shall exist **at the time when the contract phase generation is executed.**]()

**[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.]  
 (*RS\_SWCT\_03110*)

This aspect is explained in more detail in chapter 4.2.5.

Note that the mapping classes instead of direct associations have been introduced for process reasons: It allows maintaining 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.

Of course, if `ImplementationDataTypes` are generated from existing `ApplicationDataTypes` it is expected that they will be automatically compatible.](*RS\_SWCT\_03216, RS\_SWCT\_03217*)

Please note that the compatibility between an `ApplicationDataType` and an `ImplementationDataType` mapped onto each other is clarified in chapter 6.2.5.

Furthermore, the various mappings are aggregated in a container `DataTypeMappingSet` for easier maintenance in artifacts.

<b>Class</b>	<b>DataTypeMappingSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	This class represents a list of mappings between <code>ApplicationDataTypes</code> and <code>ImplementationDataTypes</code> . In addition, it can contain mappings between <code>ImplementationDataTypes</code> and <code>ModeDeclarationGroups</code> . <b>Tags:</b> atp.recommendedPackage=DataTypeMappingSets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> , <a href="#">Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataTypeMap	<a href="#">DataTypeMap</a>	*	aggr	This is one particular association between an <code>ApplicationDataType</code> and its <code>AbstractImplementationDataType</code> .
modeRequestTypeMap	<a href="#">ModeRequestTypeMap</a>	*	aggr	This is one particular association between an <code>ModeDeclarationGroup</code> and its <code>AbstractImplementationDataType</code> .

**Table 5.5: 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 some Software Component or Basic Software Module Descriptions.

How to organize [DataTypeMappingSet](#)s 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 [5.85](#).

**[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 [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

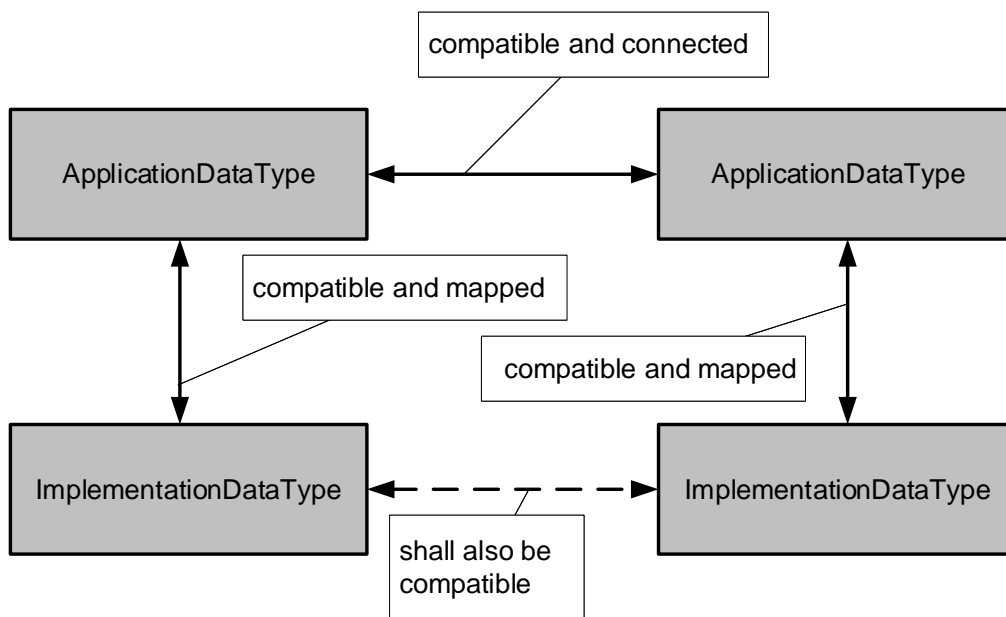
a more simple type system on the implementation level, than on the application model level.

- The same `ApplicationDataTypes` may be mapped to different `ImplementationDataTypes` for different ECUs. This scenario allows to choose the implementation data types according to the needs of specific ECUs.
- The same `ApplicationDataTypes` may be mapped to different `ImplementationDataTypes` even in the scope of a single ECU (more exactly speaking, a single RTE), but only for different `AtomicSwComponentTypes` (see [`constr_1004`]).

This improves the portability of software components which were developed independently or are ported between ECUs.

]()

**[constr\_1004] Mapping of `ApplicationDataTypes` in the scope of single `AtomicSwComponentTypes`** [In the scope of `AtomicSwComponentType.internalBehavior.dataTypeMapping`, each `ApplicationDataType` shall be mapped to exactly one `ImplementationDataType` **at the time when the contract phase generation is executed.**]()



**Figure 5.2: Compatibility of Data Types**

**[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) **at the time when the contract phase generation is executed.**]()

This constraint is visualized in figure 5.2.

**[constr\_1636] Mapping of data types that represent an Optional Element Structure** [An `ApplicationRecordDataType` with at least one `element` where attribute `isOptional` is set to `True` shall only be mapped to an `ImplementationDataType` that fulfills the structural requirements to represent an `Optional Element Structure` (see [TPS\_SWCT\_01774]) **at the time when the contract phase generation is executed.**]()

### 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 [11]).

**[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.]()

Detailed explanations about the semantics of meta-class `SwDataDefProps` can be found in chapter 5.4.

**[constr\_1006] applicable data categories** [Table 5.6 defines the **only** applicable `category`s depending on specific model elements related to data definition properties **at any time in the workflow.**]()

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 `AutosarDataType.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 5.6. Some `category`s are also applied to sub-elements of the type system (column “Applicable to ...” in table 5.6). This is explained in more detail in the following sections.

Please note that the column “RTE + BSW” of table 5.6 is only applicable for `category`s that are relevant either for `ImplementationDataTypes` and/or the aspect of measurement and calibration in `McDataInstance`.

Category	Applicable to ...													Use Case				Description
	ApplicationArrayType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValuesSpecification	ApplicationRuleBasedValuesSpecification	ImplementationDataType	ImplementationDataTypeElement	SwServiceArg	SwSystemconst	McDataInstance	Calibration	Measurement	Communication Port Interfaces	RTE + BSW		
<b>VALUE</b>			x	x	x	x		x	x	x <sup>3</sup>	x	x	x	x	x	x	Contains a single value.	
<b>VAL_BLK</b>			x	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).	
<b>DATA_REFERENCE</b>								x	x	x					x <sup>4</sup>	x	Contains an address of another <i>DataPrototype</i> (whose type is given via <i>SwDataDefProps.swPointerTargetProps</i> ).	
<b>FUNCTION_REFERENCE</b>								x	x	x						x	Contains an address of a function prototype (whose signature is given via <i>SwDataDefProps.swPointerTargetProps.functionPointerSignature</i> ).	
<b>TYPE_REFERENCE</b>								x	x	x					x	x	The element is defined via reference to another data type (via <i>SwDataDefProps.implementationDataType</i> ).	
<b>STRUCTURE</b>	x			x	x			x	x			x	x	x	x	x	Holds one or several further elements which can have different <i>AutosarDataTypes</i> . The underlying elements are defined in the same manner as normal data except for the association to <i>SwAddrMethod</i> : This has to be the same for all underlying elements. Corresponds to a Record if used in the application domain.	
<b>UNION</b>								x	x			x	x	x	x	x	Can hold values of different data types. It is similar to <b>STRUCTURE</b> except that all of its members start at the same location in memory. A <b>UNION</b> data prototype can contain only one of its elements at a time. The size of the <b>UNION</b> is at least the size of the largest member. Please find more information in [TPS_SWCT_01700].	
<b>ARRAY</b>	x			x	x			x	x			x	x	x	x	x	An array of sub-elements which are of the same type.	
<b>BIT</b>												x	x	x		x	One or several bits within a host variable, which are treated as an own data object.	



<sup>3</sup>This option has **very few** valid use cases, e.g. for defining a function pointer in native C notation, for example: `int (*SwCluC_BManif_VoidFncPtrType)(void);`

<sup>4</sup>[constr\_1295] applies!



Category	Applicable to ...													Use Case				Description
	ApplicationArrayDataType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValueSpecification	ApplicationRuleBasedValueSpecification	ImplementationDataType	ImplementationDataTypeElement	SwServiceArg	SwSystemconst	McDataInstance	Calibration	Measurement	Communication Port Interfaces	RTE + BSW		
<b>HOST</b>											x	x	x		x		A <b>HOST</b> data type is like a simple <b>VALUE</b> , but it is used for packed bit definition. That means it can host several <b>BIT</b> variables which have their own description and measurement access.	
<b>STRING</b>			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).	
<b>BOOLEAN</b>			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.	
<b>COM_AXIS</b>			x	x	x	x	x					x	x		x		An axis definition as separate calibration parameter which can be referenced by any <b>CURVE</b> , <b>MAP</b> , <b>CUBOID</b> , <b>CUBE_4</b> , and <b>CUBE_5</b> . 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 <b>CURVES</b> , <b>MAPS</b> , <b>CUBOIDS</b> , <b>CUBE_4s</b> , and <b>CUBE_5s</b> .	
<b>RES_AXIS</b>			x	x	x	x	x					x	x		x		A <b>RES_AXIS</b> (rescale axis) is also a shared axis like <b>COM_AXIS</b> , the difference is that this kind of axis can be used for rescaling. Note that the <b>RES_AXIS</b> is by nature a <b>CURVE</b> 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 <b>COM_AXIS</b> , it can compress a huge range to a non-linear distributed axis points thus retaining the required accuracy.	
<b>CURVE</b>			x	x	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 <b>CURVE</b> 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.	





Category	Applicable to ...											Use Case				Description	
	ApplicationArrayDataType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValueSpecification	ApplicationRuleBasedValueSpecification	ImplementationDataType	ImplementationDataElement	SwServiceArg	SwSystemconst	McDataInstance	Calibration	Measurement	Communication Port Interfaces	RTE + BSW	
MAP			x	x	x	x	x					x	x		x		<p>Calibration parameter with two input values and one output value. That means output values can be defined depending on the input values.</p> <p>The granularity of implemented functionality can be changed by using different number of axis points for y- and x-axis. A <b>MAP</b> has always two input axes and one output axis.</p> <p>The output axis is a characteristic of the <b>MAP</b> and every time present but the input axes can be defined within the <b>MAP</b> definition or separately.</p>
CUBOID			x	x	x	x	x					x	x		x		<p>Calibration parameter with three input values and one output value. That means output values can be defined depending on the input values.</p> <p>The granularity of implemented functionality can be changed by using different number of axis points for the input axes. A <b>CUBOID</b> has always three input axes and one output axis.</p> <p>The output axis is a characteristic of the <b>CUBOID</b> and every time present but the input axes can be defined within the <b>CUBOID</b> definition or separately.</p>
CUBE_4			x	x	x	x	x					x	x		x		<p>Calibration parameter with four input values and one output value. That means output values can be defined depending on the input values.</p> <p>The granularity of implemented functionality can be changed by using different number of axis points for the input axes. A <b>CUBE_4</b> has always four input axes and one output axis.</p> <p>The output axis is a characteristic of the <b>CUBE_4</b> and every time present but the input axes can be defined within the <b>CUBE_4</b> definition or separately.</p>
CUBE_5			x	x	x	x	x					x	x		x		<p>Calibration parameter with five input values and one output value. That means output values can be defined depending on the input values.</p> <p>The granularity of implemented functionality can be changed by using different number of axis points for the input axes. A <b>CUBE_5</b> has always five input axes and one output axis.</p> <p>The output axis is a characteristic of the <b>CUBE_5</b> and every time present but the input axes can be defined within the <b>CUBE_5</b> definition or separately.</p>
MACRO										x						x	This represents an argument to a C macro.

 Table 5.6: 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** [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**)

This aspect is further explained in Figure 5.3.

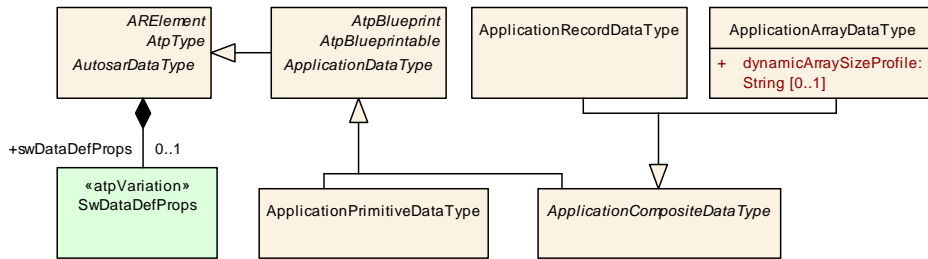


Figure 5.3: Basic Meta-Model for **ApplicationDataType**

Attributes of SwDataDefProps	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
<b>additionalNativeTypeQualifier</b>																
<b>annotation</b>	x	x	x	*	*	*	*	*	*	*	*	*	*	*	*	*
<b>baseType</b>																
<b>compuMethod</b>	x			0..1	0..1				0..1			0..1	0..1	0..1	0..1	0..1
<b>dataConstr.dataConstrRule.physConstrs</b>	x	x	x	0..1	0..1		0..1		0..1			0..1	0..1	0..1	0..1	0..1
<b>dataConstr.dataConstrRule.internalConstrs</b>	x	x	x	d/c <sup>5</sup>	d/c		d/c		d/c			d/c	d/c	d/c	d/c	d/c
<b>displayFormat</b>	x	x	x	0..1	0..1		0..1	0..1	0..1			0..1	0..1	0..1	0..1	0..1
<b>displayPresentation</b>	x	x	x	0..1	0..1		0..1			0..1	0..1	0..1	0..1	0..1	0..1	0..1
<b>implementationDataType</b>																

<sup>5</sup>don't care



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	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	0..1
swAlignment																	
swBitRepresentation																	
swCalibrationAccess	x	x		0..1	0..1	0..1	0..1	0..1	0..1	1	1	1	1	1	1	1	1
swCalprmAxisSet	x									1	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	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	0..1
swIsVirtual																	
swPointerTargetProps																	
swRecordLayout	x			0..1	0..1 <sup>6</sup>			0..1		1	1	1	1	1	1	1	1
swRefreshTiming	x			0..1	0..1			0..1	0..1								
swTextProps	x							1									
swValueBlockSize	x				1												
swValueBlockSizeMult	x				1												
unit	x			0..1	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	0..1
<b>Other Attributes below the Root Element</b>																	
element: ApplicationRecordElement	x	x	x				1..*										
element: ApplicationArrayElement	x	x	x				1										
ApplicationArrayElement.array- SizeSemantics	x						0..1										
ApplicationArrayElement. maxNumberOfElements	x						1										

**Table 5.7: Allowed Attributes vs. category for ApplicationDataTypes**

<b>Class</b>	<b>ApplicationPrimitiveDataType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	A primitive data type defines a set of allowed values. <b>Tags:</b> atp.recommendedPackage=ApplicationDataTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">ApplicationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 5.8: ApplicationPrimitiveDataType**

<sup>6</sup>This is required by [TPS\_SWCT\_01179].

<b>Class</b>	<b>ApplicationCompositeDataType</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	Abstract base class for all application data types composed of other data types.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">ApplicationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">ApplicationArrayDataType</a> , <a href="#">ApplicationRecordDataType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 5.9: ApplicationCompositeDataType**

**[TPS\_SWCT\_01241] Applicable [category](#)s 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 **at any time in the workflow** are listed as an overview in table 5.7.] ()

This list makes use of the [SwDataDefProps](#) and other meta-model elements which are explained in detail in the further sections of this chapter.

## 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 [Application-PrimitiveDataType](#) is actually represented by “integer” numbers.

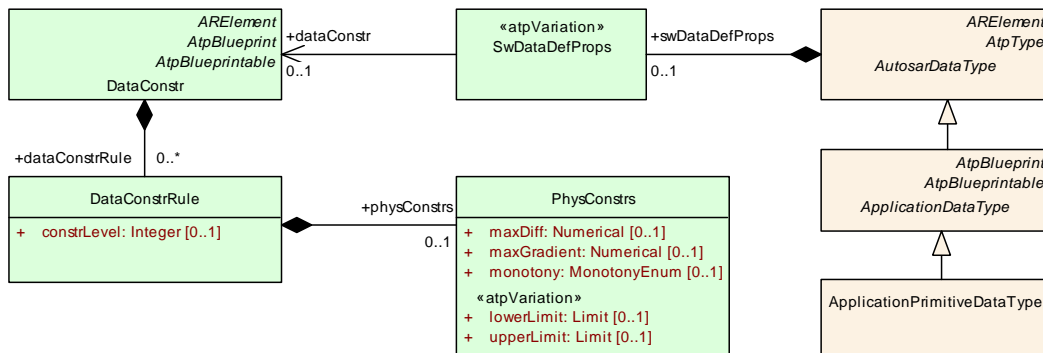


Figure 5.4: Specification of Physical Limits

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`.

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 `invalidValue` can be defined which is used by the RTE in order to initialize values of `VariableDataPrototypes/ParameterDataPrototypes` defined locally in a software-component.

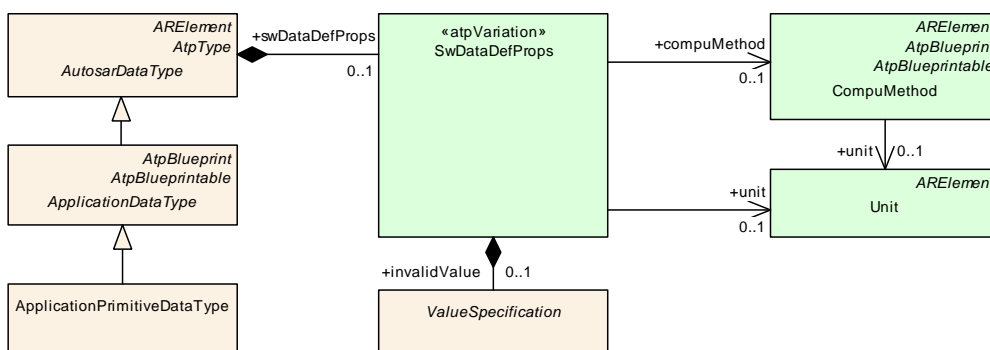
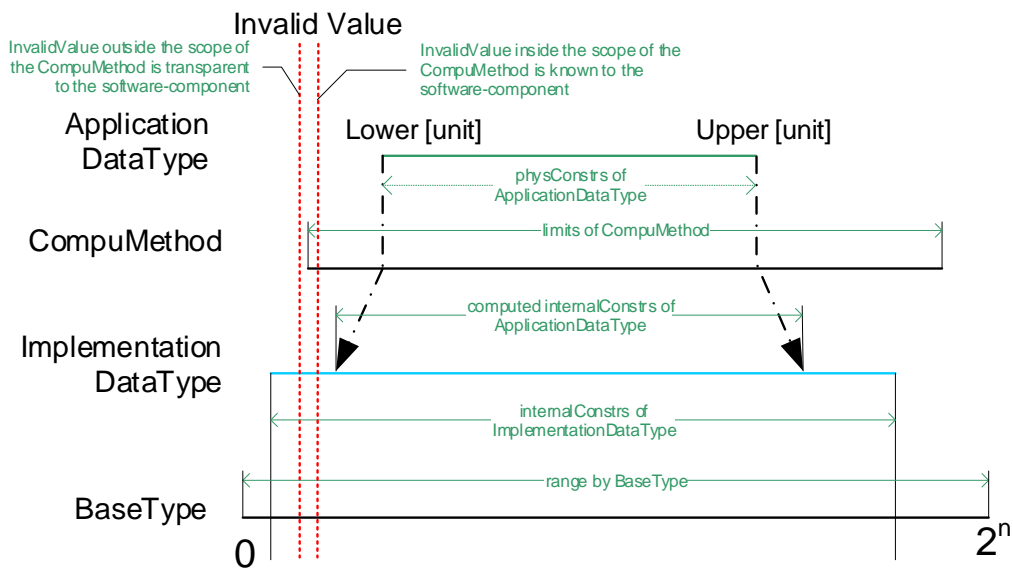


Figure 5.5: Some Properties of `ApplicationPrimitiveDataTypes`

Figure 5.6 illustrates the relationship between the data constraints for `ApplicationDataType`, `CompuMethod`, `ImplementationDataType`, `BaseType` and also the `invalidValue` for the case of an entirely linear or rational conversion.

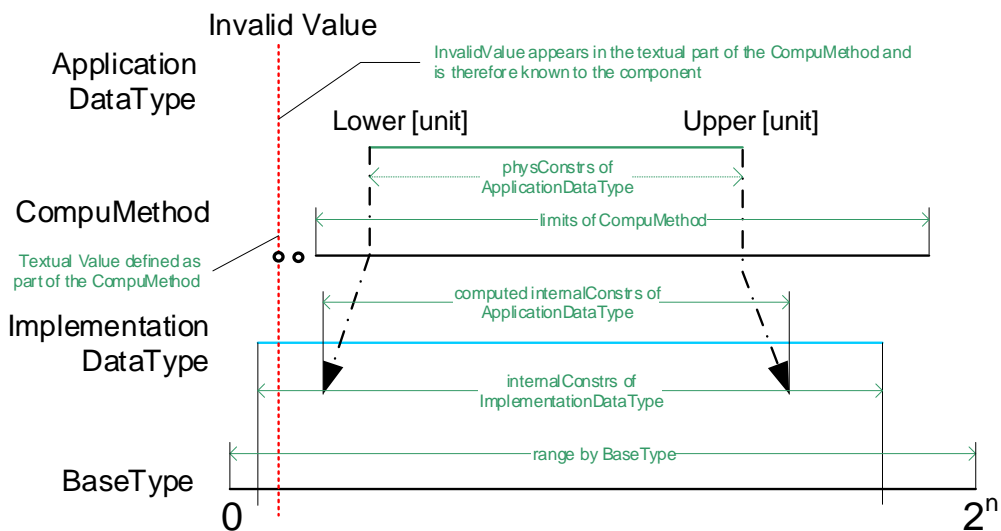
Please note that Figure 5.6 is only applicable for linear and rational `CompuMethods`.





**Figure 5.6: Value ranges and invalid values for linear and rational [CompuMethod](#)**

Figure 5.7 and Figure 5.8 depict a similar situation for the case of mixed [CompuMethods](#) where the [invalidValue](#) is defined in the discrete part of a [CompuMethod](#).

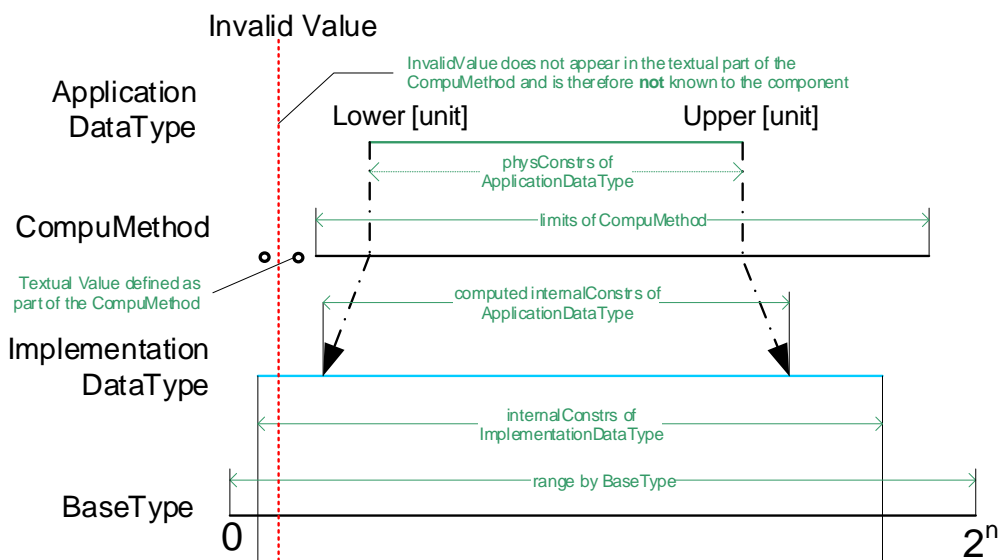


**Figure 5.7: Value ranges and invalid values with discrete [invalidValue](#) defined inside the scope of the [CompuMethod](#)**

Figure 5.7 sketches a case where a `CompuMethod` has a linear and a discrete part and the `invalidValue` is defined by means of one value that is defined in the discrete part of the `CompuMethod`.

As mentioned by [TPS\_SWCT\_01834], the `invalidValue` shall be defined in the physical domain in this case. In other words, the `invalidValue` shall be defined by a symbol according to [TPS\_SWCT\_01432].

As a consequence of the definition of an `invalidValue` **inside**, the scope of a mixed `CompuMethod` the `invalidValue` is visible to the software-component.



**Figure 5.8: Value ranges and invalid values with discrete `invalidValue` defined outside the scope of the `CompuMethod`**

Figure 5.8, on the other hand, sketches a case where a `CompuMethod` has a linear and a discrete part and the `invalidValue` is not within the defined linear interval and not defined by means of one value out of the discrete part of the `CompuMethod`.

As mentioned by [TPS\_SWCT\_01835], the `invalidValue` shall be defined in the internal domain in this case. In other words, the `invalidValue` shall be defined by a `NumericalValueSpecification`.

As a consequence of the definition of an `invalidValue` **outside** the scope of a mixed `CompuMethod`, the `invalidValue` is invisible (and therefore not accessible) to the software-component.

If an `ApplicationPrimitiveDataType` does not define `dataConstr`, then implicit constraints can be derived from physical meaning of the `ApplicationDataType`.

For example, if the data type represents a temperature the lower bound will not be able to exceed 0K.

For other physical meanings, it could be possible that the implicitly assumed limits go from  $-\text{INF}$  to  $+\text{INF}$ .

In order to avoid ambiguity regarding the values of limits it is **strongly recommended** defining a reasonable limit for `ApplicationPrimitiveDataTypes`.

#### [constr\_2544] Limits need to be consistent [

- The limits of `ApplicationDataType` shall be inside 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

[TPS\_SWCT\_01834] **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` shall be used to describe the `invalidValue` of the `ApplicationPrimitiveDataType`.](*RS\_SWCT\_03216*)

[TPS\_SWCT\_01834] means that the value of the `ApplicationValueSpecification` shall be within the bounds defined by `swDataDefProps.compuMethod.compuPhysToInternal.compuContent.compuScale.lowerLimit` or `upperLimit` or the inverse case that is based on the bounds defined by `swDataDefProps.compuMethod.compuInternalToPhys.compuContent.compuScale.lowerLimit` or `upperLimit`.

[TPS\_SWCT\_01835] **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` (that provides a value in the internal representation) shall be used to describe the `invalidValue` of the `ApplicationPrimitiveDataType`.](*RS\_SWCT\_03216*)

Because of the existence of [TPS\_SWCT\_01834] and [TPS\_SWCT\_01835], the definition of the `invalidValue` is fully specified and therefore [constr\_1221] does not apply to this case.

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.

**[TPS\_SWCT\_01760] Defining the dimension of an `ApplicationPrimitiveDataType` of category `VAL_BLK`** [An `ApplicationPrimitiveDataType` of category `VAL_BLK` that has only one dimension shall be described using the attribute `SwDataDefProps.swValueBlockSize`.

An `ApplicationPrimitiveDataType` of category `VAL_BLK` that has more than one dimension shall be described using the attribute `SwDataDefProps.swValueBlockSizeMult`.] (*RS\_SWCT\_03216*)

**[constr\_1610] Existence of `SwDataDefProps.swValueBlockSize` and `SwDataDefProps.swValueBlockSizeMult`** [Attributes `SwDataDefProps.swValueBlockSize` and `SwDataDefProps.swValueBlockSizeMult` shall not exist at the same time in the context of a given `SwDataDefProps`.

This rule shall be imposed **at the time when the contract phase generation is executed**.] ()

#### 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 discrete set 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`.

] (*RS\_SWCT\_03216*)

The relevant meta-classes in the context of `SwDataDefProps` are sketched in Figure 5.9. This includes all meta-classes that may contribute to the definition of the symbol of a `CompuScale` in C code, see [TPS\_SWCT\_01431].

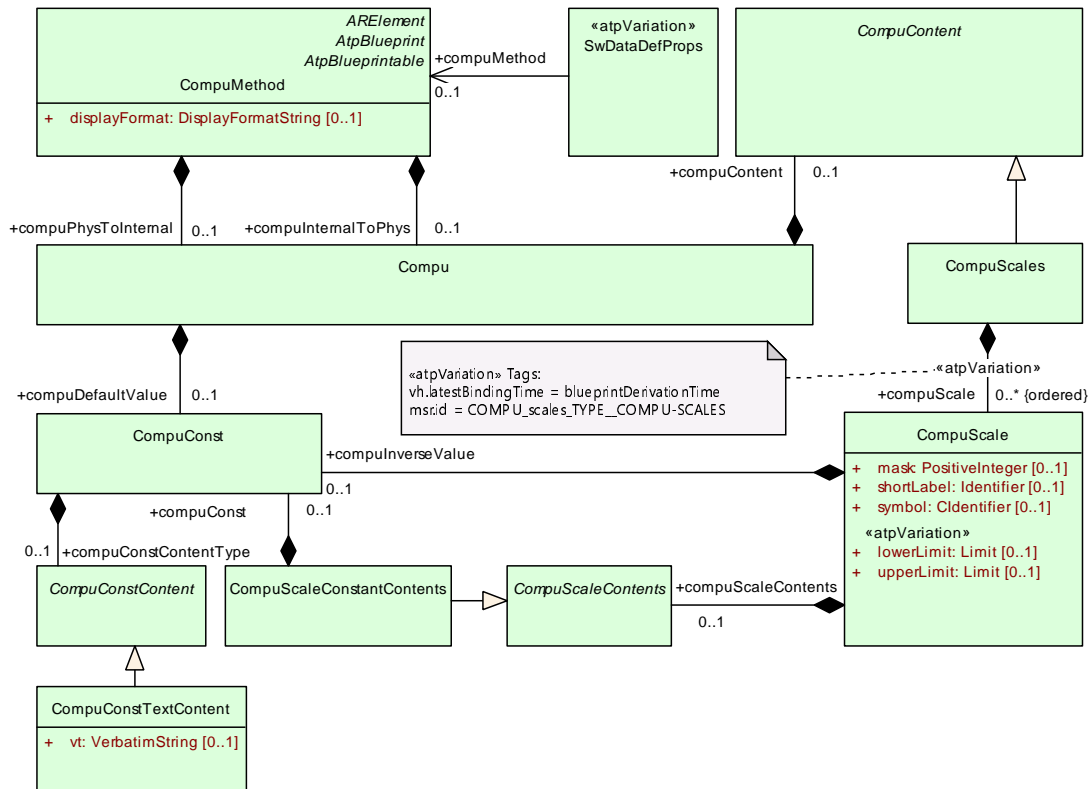


Figure 5.9: Relevant meta-classes for the specification of enumerations

An example of how an enumeration looks like in ARXML is contained in section 5.5.1.4.

### 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 similarly of other `AutosarDataTypes`. Their substructure needs a special description in order to be compatible with existing calibration techniques.] (`RS_SWCT_03216`)

[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.] (*RS\_SWCT\_03216*)

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`.

This rule shall be imposed **at the time when the contract phase generation is executed**]()

[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`.]()

[constr\_1905] **Existence of attribute SwTextProps.arraySizeSemantics** [For each `SwTextProps`, attribute `arraySizeSemantics` shall exist **at the time when the contract phase generation is executed**.]()

[constr\_1906] **Existence of attribute SwTextProps.swMaxTextSize** [For each `SwTextProps`, attribute `swMaxTextSize` shall exist **at the time when the contract phase generation is executed**.]()

[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*)

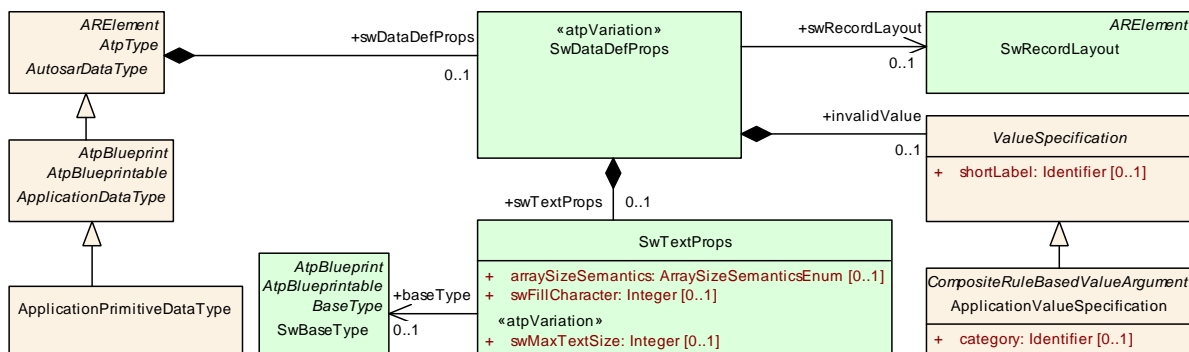


Figure 5.10: Specification of textual strings

[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`.

<b>Class</b>	<b>SwTextProps</b>			
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties			
<b>Note</b>	This meta-class expresses particular properties applicable to strings in variables or calibration parameters.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arraySize Semantics	<a href="#">ArraySizeSemantics Enum</a>	0..1	attr	This attribute controls the semantics of the arraysize for the array representing the string in an Implementation DataType.  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.
baseType	<a href="#">SwBaseType</a>	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.  <b>Tags:</b> 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 arraySize Semantics.  <b>Tags:</b> xml.sequenceOffset=40
swMaxTextSize	Integer	0..1	attr	Specifies the maximum text size in characters. Note the size in bytes depends on the encoding in the corresponding baseType.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

**Table 5.10: SwTextProps**

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 5.10).

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.

**Listing 5.1: Example for the definition of a string `ApplicationPrimitiveDataType`**

```

<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 BASE="default" DEST="SW-BASE-TYPE">BaseTypes/
            MyTextBaseType</BASE-TYPE-REF>
        </SW-TEXT-PROPS>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
      <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 BASE="default" DEST="SW-RECORD-LAYOUT">
        RecordLayouts/StringDescriptor</SW-RECORD-LAYOUT-REF>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-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.

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`.

#### Listing 5.2: Example for the definition of a `SwRecordLayout` for an `ApplicationPrimitiveDataType` of `category STRING`

```

<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-GROUP>
      <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>

```

The next listing describes the data type of one character within the string data type.

**Listing 5.3: Example for the definition of the character data type of a string [ImplementationDataType](#)**

```

<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/uint8BT</BASE-TYPE-REF
        >
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</IMPLEMENTATION-DATA-TYPE>

```

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 5.4: Example for the definition of a string [ImplementationDataType](#)**

```

<IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>MyImplementationStringType</SHORT-NAME>
  <CATEGORY>STRUCTURE</CATEGORY>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <INVALID-VALUE>
          <RECORD-VALUE-SPECIFICATION>
            <FIELDS>
              <NUMERICAL-VALUE-SPECIFICATION>
                <VALUE>3</VALUE>
              </NUMERICAL-VALUE-SPECIFICATION>
              <ARRAY-VALUE-SPECIFICATION>
                <ELEMENTS>
                  <NUMERICAL-VALUE-SPECIFICATION>
                    <VALUE>105</VALUE>
                  </NUMERICAL-VALUE-SPECIFICATION>
                  <NUMERICAL-VALUE-SPECIFICATION>
                    <VALUE>110</VALUE>
                  </NUMERICAL-VALUE-SPECIFICATION>
                  <NUMERICAL-VALUE-SPECIFICATION>
                    <VALUE>118</VALUE>
                  </NUMERICAL-VALUE-SPECIFICATION>
                </ELEMENTS>
              </ARRAY-VALUE-SPECIFICATION>
            </RECORD-VALUE-SPECIFICATION>
          </INVALID-VALUE>
        </SW-DATA-DEF-PROPS-CONDITIONAL>
      </SW-DATA-DEF-PROPS-VARIANTS>
    </SW-DATA-DEF-PROPS>
  </IMPLEMENTATION-DATA-TYPE>

```

```

        </ARRAY-VALUE-SPECIFICATION>
    </FIELDS>
</RECORD-VALUE-SPECIFICATION>
</INVALID-VALUE>
</SW-DATA-DEF-PROPS-CONDITIONAL>
</SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
<DYNAMIC-ARRAY-SIZE-PROFILE>VSA_LINEAR</DYNAMIC-ARRAY-SIZE-PROFILE>
<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>
                </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>200</ARRAY-SIZE>
                <ARRAY-SIZE-HANDLING>ALL-INDICES-SAME-ARRAY-SIZE</ARRAY-SIZE-
                    HANDLING>
                <ARRAY-SIZE-SEMANTICS>VARIABLE-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>

```

The `ImplementationDataType` with the `shortName` `MyImplementation-StringType` (as exemplified in Listing 5.4) represents a variable-size string, i.e. the number of characters may change at run-time. Of course, it is also possible to model a fixed-size string, but in that case a size-indicator is not required.

Please note further that the size of the payload array in the definition of the `ImplementationDataType` in Listing 5.4 has been set to the value 200 in order to accommodate for the definition of `swMaxTextSize` in the definition of the corresponding

`ApplicationDataType` in combination with the fact that the value of `baseTypeEncoding` has been set to `UTF-8`.

For background, the value of attribute `SwTextProps.swMaxTextSize` shall be specified as the number of *code points* in the string.

Each *code point* will be encoded by a sequence of bytes, depending on the applicable encoding. In the case of `UTF-8`, for example, each *code point* will be encoded by up to four bytes.

On the level of `ImplementationDataType`, an array designed to hold a string consisting of *code points* encoded using `UTF-8` needs to be big enough to carry the number of *code points* (which may have been described by `SwTextProps.swMaxTextSize`) times 4 bytes.

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”).

The value element 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 5.5: Example for the definition of a string `SwBaseType`

```
<SW-BASE-TYPE>
  <SHORT-NAME>MyTextBaseType</SHORT-NAME>
  <CATEGORY>FIXED_LENGTH</CATEGORY>
  <BASE-TYPE-SIZE>8</BASE-TYPE-SIZE>
  <BASE-TYPE-ENCODING>UTF-8</BASE-TYPE-ENCODING>
</SW-BASE-TYPE>
<SW-BASE-TYPE>
  <SHORT-NAME>uint8BT</SHORT-NAME>
  <CATEGORY>FIXED_LENGTH</CATEGORY>
  <BASE-TYPE-SIZE>8</BASE-TYPE-SIZE>
</SW-BASE-TYPE>
```

The contribution of this definition of `SwBaseType` to the overall definition of a string data type is represented by the definition of the character encoding (which is set to `UTF-8`).

#### Listing 5.6: Example for the definition of the applicable `DataTypeMappingSet`

```
<DATA-TYPE-MAPPING-SET>
  <SHORT-NAME>theExample</SHORT-NAME>
  <DATA-TYPE-MAPS>
```

```

<DATA-TYPE-MAP>
  <APPLICATION-DATA-TYPE-REF BASE="default" DEST="APPLICATION-PRIMITIVE-
    -DATA-TYPE">ApplicationDataTypes/MyApplicationStringType</
    APPLICATION-DATA-TYPE-REF>
  <IMPLEMENTATION-DATA-TYPE-REF BASE="default" DEST="IMPLEMENTATION-
    -DATA-TYPE">ImplementationDataTypes/MyImplementationStringType</
    IMPLEMENTATION-DATA-TYPE-REF>
</DATA-TYPE-MAP>
</DATA-TYPE-MAPS>
</DATA-TYPE-MAPPING-SET>

```

However, there is still one important part missing, i.e. the definition of the mapping of `ApplicationPrimitiveDataType` to `ImplementationDataType` (and vice versa, see Listing 5.6).

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` 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.] (*RS\_SWCT\_03215*, *RS\_SWCT\_03216*)

Details about meta-class `ApplicationRecordDataType` are depicted in Figure 5.11.

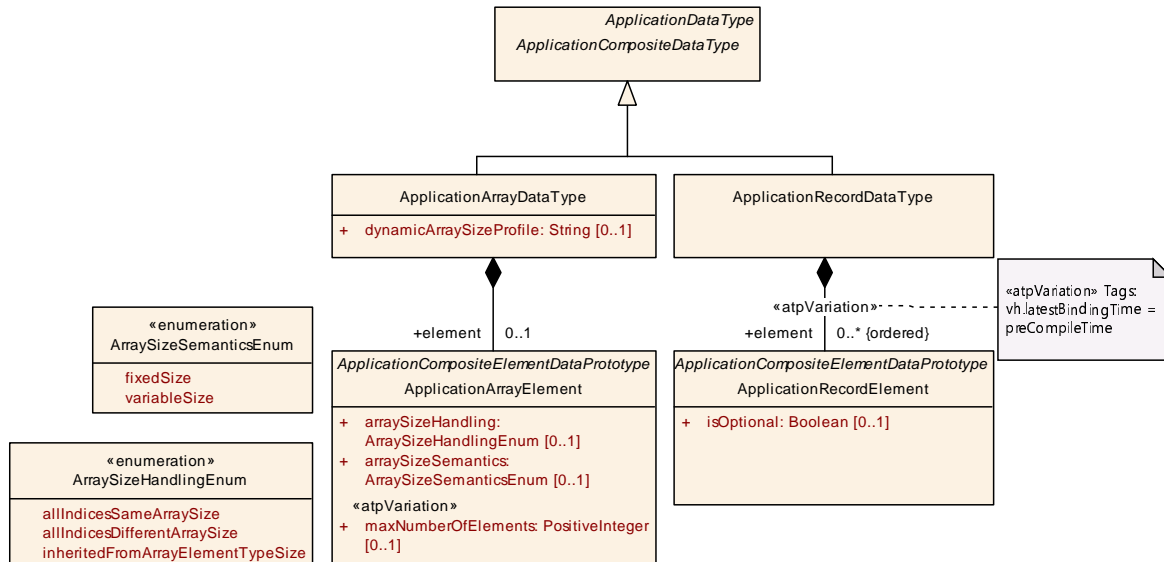


Figure 5.11: Summary of `ApplicationCompositeDataType`

### 5.2.4.2.1 ApplicationArrayDataType

[TPS\_SWCT\_01078] **Configurable array size** [The size of an `ApplicationArrayDataType` (in terms of the number of elements) is configured by means of the value of attribute `ApplicationArrayElement.maxNumberOfElements`.

For the purpose of referring to an element of an `ApplicationArrayDataType` within a software-component description, the element's `index` runs from 0 to the value of `maxNumberOfElements-1`.] (*RS\_SWCT\_03144*, *RS\_SWCT\_03215*)

<b>Class</b>	<code>ApplicationArrayDataType</code>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	An application data type which is an array, each element is of the same application data type. <b>Tags:</b> atp.recommendedPackage=ApplicationDataTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">ApplicationCompositeDataType</a> , <a href="#">ApplicationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dynamicArraySizeProfile	<a href="#">String</a>	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.
element	<a href="#">ApplicationArrayElement</a>	0..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 5.11: `ApplicationArrayDataType`

**[constr\_1907] Existence of attribute `ApplicationArrayDataType.element`**  
 [For each `ApplicationArrayDataType`, the aggregation of `ApplicationArrayElement` in the role `element` shall exist **at the time when the RTE is generated.**]()

<b>Class</b>	<b>ApplicationArrayElement</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	Describes the properties of the elements of an application array data type.			
<b>Base</b>	<i>ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arraySizeHandling	<a href="#">ArraySizeHandlingEnum</a>	0..1	attr	The way how the size of the array is handled.
arraySizeSemantics	<a href="#">ArraySizeSemanticsEnum</a>	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
indexDataType	<a href="#">ApplicationPrimitiveDataType</a>	0..1	ref	This reference can be taken to assign a CompuMethod of category TEXTTABLE to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name.
maxNumberOfElements	PositiveInteger	0..1	attr	The maximum number of elements that the array can contain. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 5.12: ApplicationArrayElement**

Please note that the information about the number of elements of a specific `ApplicationArrayDataType` 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 `ApplicationArrayDataType` 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\\_03215](#), [RS\\_SWCT\\_03144](#))

<b>Enumeration</b>	<b>ArraySizeSemanticsEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes





Enumeration	ArraySizeSemanticsEnum
Note	This type controls how the information about the number of elements in an ApplicationArrayDataType is to be interpreted.
Literal	Description
fixedSize	This means that the ApplicationArrayDataType will always have a fixed number of elements. Tags:atp.EnumerationLiteralIndex=0
variableSize	This implies that the actual number of elements in the ApplicationArrayDataType might vary at run-time. The value of arraySize represents the maximum number of elements in the array. Tags:atp.EnumerationLiteralIndex=1

**Table 5.13: 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`.]()

Enumeration	ArraySizeHandlingEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes
Note	This enumeration defines different ways to handle the sizes of variable size arrays.
Literal	Description
allIndicesDifferent ArraySize	All elements of the variable size array may have different sizes. Tags:atp.EnumerationLiteralIndex=0
allIndicesSame ArraySize	All elements of the variable size array have the same size. Tags:atp.EnumerationLiteralIndex=1
inheritedFromArray ElementTypeSize	The size of all dimensions of the variable size array is determined by the size of the contained array element. Tags:atp.EnumerationLiteralIndex=2

**Table 5.14: ArraySizeHandlingEnum**

#### 5.2.4.2.1.1 Variable Size Array

**[TPS\_SWCT\_01604] Enable Size Indicator** [To enable the RTE's ability to consider the number of valid elements inside a `Variable-Size Array DataType` the `ApplicationArrayDataType.dynamicArraySizeProfile` of `ApplicationArrayDataType` and `ApplicationArrayElement.arraySizeHandling` shall be set.]([RS\\_SWCT\\_03181](#))

**[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](#))

**[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 coherent relations 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 `ApplicationArrayElement.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** the following conditions **at the time when the contract phase generation is executed**:

- 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** 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** 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** 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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** 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** 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** 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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** 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** 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** 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`.

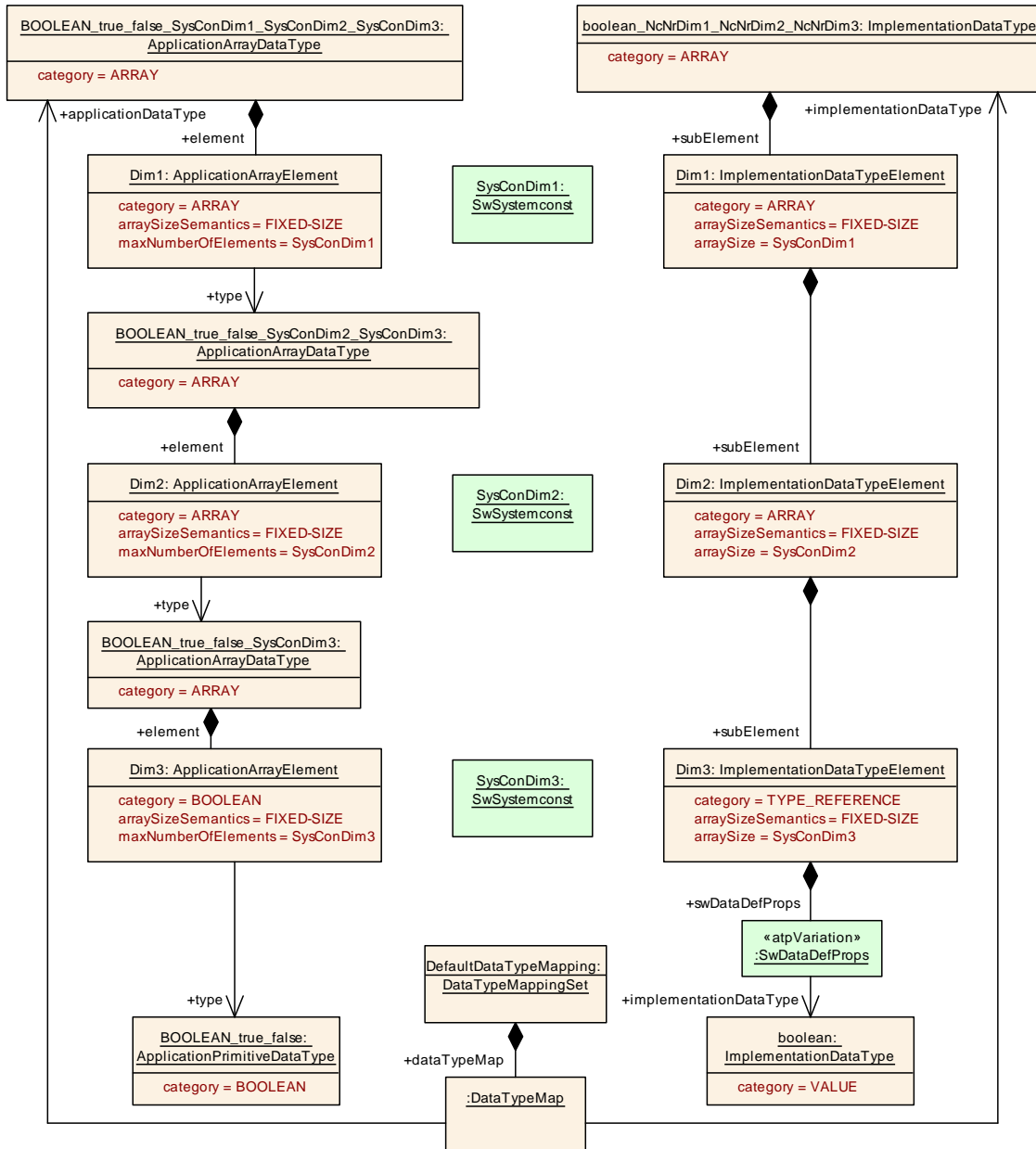
This rule shall be imposed **at the time when the contract phase generation is executed.** |()

For examples, see Appendix E.1.

#### 5.2.4.2.1.2 Multi-Dimensional Arrays

Figure 5.12 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.



**Figure 5.12: Example of a three-dimensional array type**

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](#)].

**[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`.

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` to `ApplicationArrayDataType` references.] ([RS\\_SWCT\\_03215](#), [RS\\_SWCT\\_03216](#))

#### 5.2.4.2.1.3 Index Data Type

The usage of an array represents an elegant way to group data with identical properties. This allows for an easy processing of the same functionality by iterating over the array elements.

From a functional point of view, however, each array element may have a distinct meaning that could be visible to the application software. To create this visibility, it is possible to take advantage of an existing mechanism: `CompuMethods` of category TEXT-TABLE.

**[TPS\_SWCT\_01699] Usage of `ApplicationArrayElement.indexDataType`** [The primary use case of the attribute `ApplicationArrayElement.indexDataType` is the creation of composite data type mappings or the description of measurement and calibration. Furthermore, the information could be used for documentation purposes.] ([RS\\_SWCT\\_03230](#))

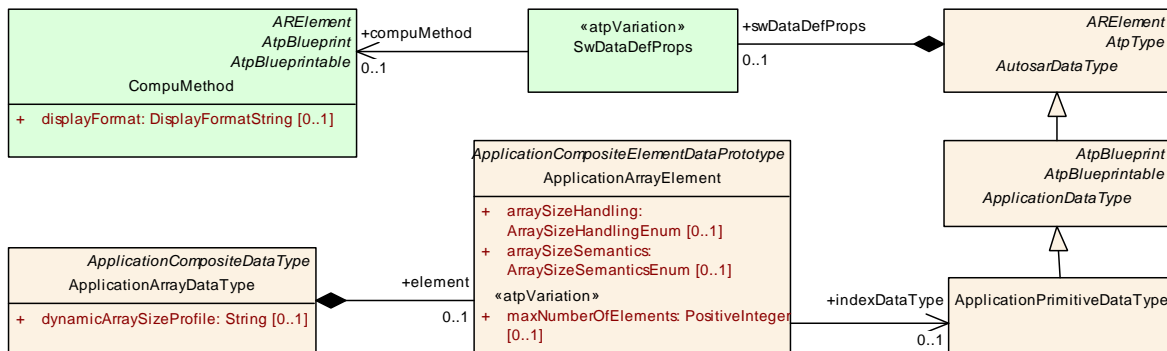


Figure 5.13: Modeling of the `ApplicationArrayElement.indexDataType`

**[constr\_1438]** `ApplicationArrayElement.indexDataType` needs to refer to a `CompuMethod` of category `TEXTTABLE` [The reference `ApplicationArrayElement.indexDataType` shall only point to an `ApplicationPrimitiveDataType` that in turn refers to a `CompuMethod` of category `TEXTTABLE`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1440]** Size of the `CompuMethod` of category `TEXTTABLE` referenced by `ApplicationArrayElement.indexDataType` [The interval defined by the `CompuScales` contained in the `CompuMethod` referenced by `ApplicationArrayElement.indexDataType` shall start at 0 and include all integer values until `ApplicationArrayElement.maxNumberOfElements - 1`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1439]** Requirements on `ApplicationArrayElement` if attribute `indexDataType` exists [If `ApplicationArrayElement.indexDataType` exists then the attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `fixedSize` and attribute `arraySizeHandling` shall not exist **at the time when the contract phase generation is executed.**]()

Listing 5.7 exemplifies the definition of an `indexDataType`.

#### Listing 5.7: Example for array index data type

```

<APPLICATION-ARRAY-DATA-TYPE>
  <SHORT-NAME>CylinderArray</SHORT-NAME>
  <ELEMENT>
    <SHORT-NAME>CylinderArrayElement</SHORT-NAME>
    <ARRAY-SIZE-SEMANTICS>FIXED-SIZE</ARRAY-SIZE-SEMANTICS>
    <INDEX-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">
      myIndexDataType</INDEX-DATA-TYPE-REF>
    </ELEMENT>
  </APPLICATION-ARRAY-DATA-TYPE>
<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>myIndexDataType</SHORT-NAME>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
    <SW-DATA-DEF-PROPS-CONDITIONAL>
  
```



```

        <COMPU-METHOD-REF DEST="COMPU-METHOD">cylinders</COMPU-METHOD-REF>
    </SW-DATA-DEF-PROPS-CONDITIONAL>
</SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-DATA-TYPE>
    
```

Listing 5.8 contains an example of a `CompuMethod` eligible for an `indexDataType`.

#### Listing 5.8: Example for a compu method used by an array index data type

```

<COMPU-METHOD>
  <SHORT-NAME>cylinders</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>Cylinder1</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>Cylinder2</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">2</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder3</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">3</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">3</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder4</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
    </COMPU-SCALES>
  </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
    
```

#### 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)

<b>Class</b>	<b>ApplicationRecordDataType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	An application data type which can be decomposed into prototypes of other application data types. <b>Tags:</b> atp.recommendedPackage=ApplicationDataTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">ApplicationCompositeDataType</a> , <a href="#">ApplicationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
element (ordered)	<a href="#">ApplicationRecordElement</a>	*	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>ApplicationrecordData</code> Type.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 5.15: ApplicationRecordDataType**

[constr\_1908] Existence of attribute `ApplicationRecordDataType.element` [For each `ApplicationRecordDataType`, the aggregation of `ApplicationRecordElement` in the role `element` shall exist **at the time when the RTE is generated.**]( )

<b>Class</b>	<b>ApplicationRecordElement</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	Describes the properties of one particular element of an application record data type.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">ApplicationCompositeElementDataPrototype</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing <code>ApplicationRecordElement</code> as optional. This means the that, at runtime, the <code>ApplicationRecordElement</code> may or may not have a valid value and shall therefore be ignored.  The underlying runtime software provides means to set the <code>ApplicationRecordElement</code> as not valid at the sending end of a communication and determine its validity at the receiving end.

**Table 5.16: ApplicationRecordElement**

[TPS\_SWCT\_01771] Definition of optional elements on the level of `ApplicationDataType` [The modeling approach for the definition of optional elements on the level of `ApplicationDataType` is to set the attribute `ApplicationRecordElement.isOptional` to the value `True`.

If the attribute is not set or set to the value `False` then the respective `ApplicationRecordElement` shall be considered mandatory.](RS\_SWCT\_03320)

## 5.2.5 Implementation Data Type

### 5.2.5.1 Overview

[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)

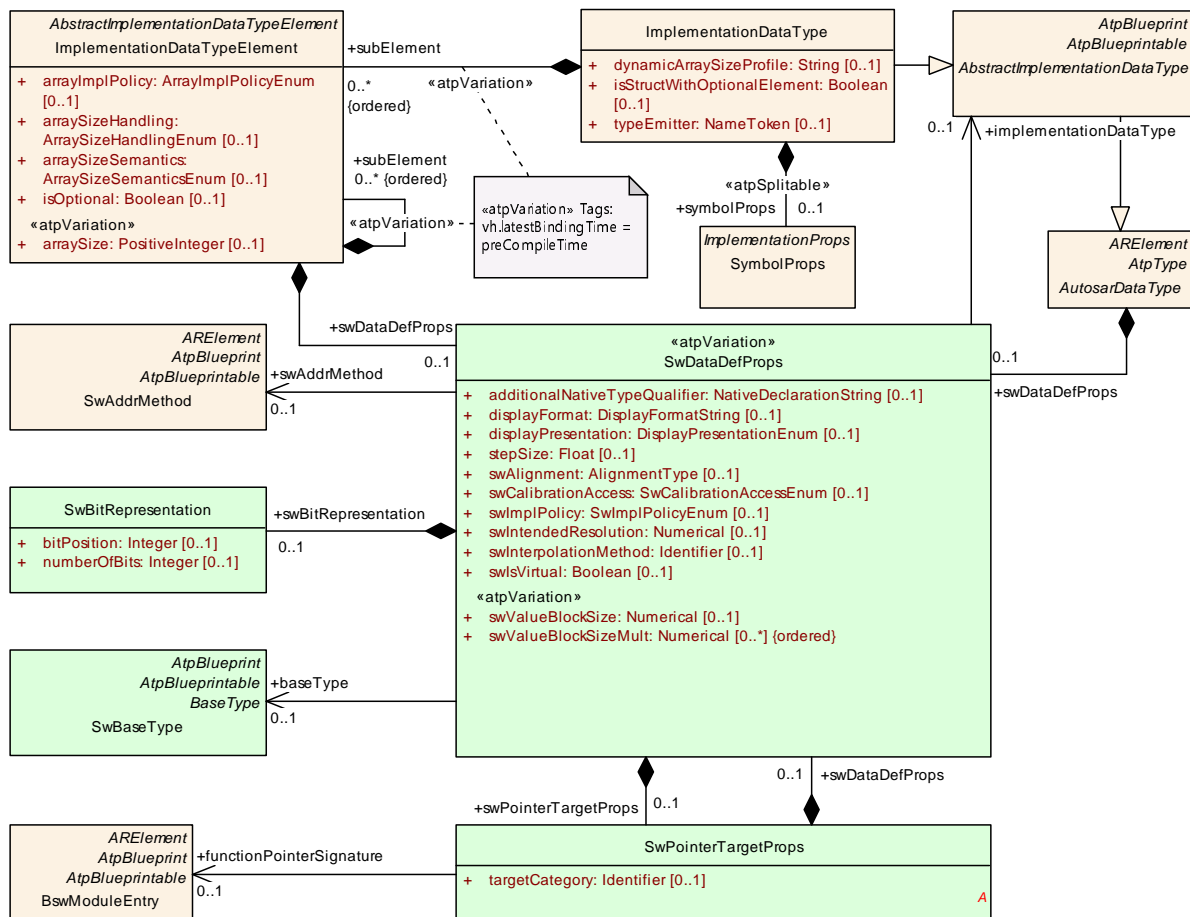


Figure 5.14: **SwDataDefProps** used in the context of **ImplementationDataType**

The allowed existence and multiplicity of all the attributes of **SwDataDefProps** and other properties depend on the **category** of the **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`

at the time when the contract phase generation is executed.>()

[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 5.17.

This rule shall be applied at any time in the workflow.>()

Attributes of SwDataDefProps	Root Element				Attribute Existence per Category						
	ImplementationDataType	ImplementationDataTypeElement	SwPointerTargetProps	SwServiceArg	VALUE	DATA_REFERENCE	FUNCTION_REFERENCE	TYPE_REFERENCE	STRUCTURE	UNION	ARRAY
<code>additionalNativeTypeQualifier</code>	x	x	x	x	0..1	0..1	0..1	0..1	0..1	0..1	0..1
<code>annotation</code>	x	x	x	x	*	*	*	*	*	*	*
<code>baseType</code>	x	x	x	x	1						
<code>compuMethod</code>	x	x	x	x	0..1			0..1			
<code>dataConstr.dataConstrRule.physConstrs</code>	x	x	x	x	d/c <sup>7</sup>			d/c			d/c
<code>dataConstr.dataConstrRule.internalConstrs</code>	x	x	x	x	0..1			0..1			0..1
<code>displayFormat</code>	x	x			0..1				0..1	0..1	0..1
<code>displayPresentation</code>	x	x			0..1						0..1
<code>implementationDataType</code>	x	x	x	x				1			
<code>invalidValue</code>	x	x	x		0..1			0..1	0..1 <sup>8</sup>		0..1 <sup>9</sup>
<code>stepSize</code>	x	x			0..1						



<sup>7</sup>don't care

<sup>8</sup>There 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.

<sup>9</sup>This 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.



Attributes of SwDataDefProps	Root Element				Attribute Existence per Category						
	ImplementationDataType	ImplementationDataTypeElement	SwPointerTargetProps	SwServiceArg	VALUE	DATA_REFERENCE	FUNCTION_REFERENCE	TYPE_REFERENCE	STRUCTURE	UNION	ARRAY
swAddrMethod	x	x	x		0..1	0..1	0..1	0..1	0..1	0..1	0..1
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	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											
swValueBlockSizeMult											
unit											
valueAxisDataType											
<b>Other Attributes</b>											
subElement: ImplementationDataTypeElement	x	x							1..*	1..*	1
subElement.arraySizeSemantics	x	x									0..1
subElement.arraySize	x	x									1

Table 5.17: Allowed Attributes vs. category for ImplementationDataType

**[constr\_1694] Allowed target of SwDataDefProps.implementationDataType**  
 [The reference SwDataDefProps.implementationDataType shall only refer to an ImplementationDataType. Any other subclass of AbstractImplementationDataType is not supported as a reference target.

This rule shall be applied **at the time when the contract phase generation is executed.** |()

This list makes use of the `SwDataDefProps` and other meta-model elements which are explained in detail in the further sections of this chapter.

Regulations regarding the applicable `categorys` for attribute `ImplementationDataType.swDataDefProps.compuMethod` can be found in [constr\_1158] inside section 5.5.1.3.2.

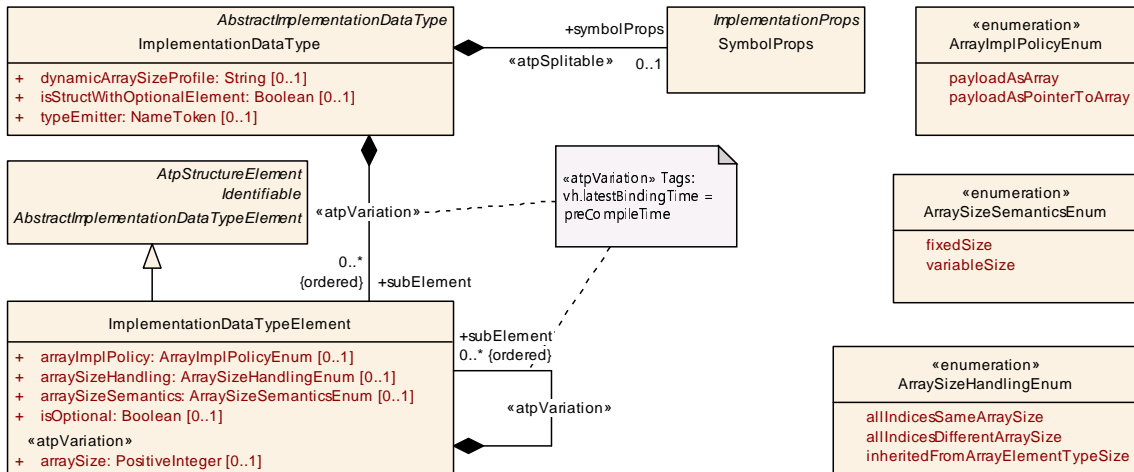


Figure 5.15: `ImplementationDataType` overview

[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 5.15. If a specific `ImplementationDataType` is supposed to define a composite data type, the `ImplementationDataType` aggregates `ImplementationDataTypeElements`.

<b>Class</b>	<i>AbstractImplementationDataType</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
<b>Note</b>	This meta-class represents an abstract base class for different flavors of <code>ImplementationDataType</code> .			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>AutosarDataType</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Subclasses</b>	<i>ImplementationDataType</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





<b>Class</b>	<b>AbstractImplementationDataType</b> (abstract)			
-	-	-	-	-

**Table 5.18: AbstractImplementationDataType**

<b>Class</b>	<b>ImplementationDataType</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
<b>Note</b>	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. <b>Tags:</b> atp.recommendedPackage=ImplementationDataTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AbstractImplementationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dynamicArraySizeProfile	<a href="#">String</a>	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWithOptionalElement	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE.  If set to True, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	<a href="#">ImplementationDataTypeElement</a>	*	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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
symbolProps	<a href="#">SymbolProps</a>	0..1	aggr	This represents the SymbolProps for the ImplementationDataType.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=symbolProps.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\_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](#))

[TPS\_SWCT\_01253] Rules apply for the usage of the attribute [ImplementationDataType.typeEmitter](#) [The following set of values is supported for the usage of the attribute [ImplementationDataType.typeEmitter](#):

- attribute [typeEmitter](#) is NOT defined.
- attribute [typeEmitter](#) is set to “RTE”.
- attribute [typeEmitter](#) is set to the name of a header file.

- attribute `typeEmitter` is set to anything else.

|(RS\_SWCT\_03217)

The consequence of setting the value of `typeEmitter` is explained in [2].

The usage of `ImplementationDataTypes` within an `AnyInstanceRef` is described in detail in [11].

<b>Class</b>	<b>AbstractImplementationDataTypeElement</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
<b>Note</b>	This meta-class represents the ability to act as an abstract base class for specific derived meta-classes that support the modeling of <code>ImplementationDataTypes</code> for a particular language binding.			
<b>Base</b>	<code>ARObject</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code>			
<b>Subclasses</b>	<code>ImplementationDataTypeElement</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 5.20: AbstractImplementationDataTypeElement**

<b>Class</b>	<b>ImplementationDataTypeElement</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
<b>Note</b>	<p>Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated.</p> <p>This element either consists of further subElements or it is further defined via its <code>swDataDefProps</code>.</p> <p>There are several use cases within the system of <code>ImplementationDataTypes</code> for such a local declaration:</p> <ul style="list-style-type: none"> <li>• It can represent the elements of an array, defining the element type and array size</li> <li>• It can represent an element of a struct, defining its type</li> <li>• It can be the local declaration of a debug element.</li> </ul>			
<b>Base</b>	<code>ARObject</code> , <code>AbstractImplementationDataTypeElement</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
<code>arrayImplPolicy</code>	<code>ArrayImplPolicyEnum</code>	0..1	attr	This attribute controls the implementation of the payload of an array. It shall only be used if the enclosing <code>ImplementationDataType</code> constitutes an array.
<code>arraySize</code>	<code>PositiveInteger</code>	0..1	attr	<p>The existence of this attributes (if bigger than 0) defines the size of an array and declares that this <code>ImplementationDataTypeElement</code> represents the type of each single array element.</p> <p><b>Stereotypes:</b> <code>atpVariation</code> <b>Tags:</b> <code>vh.latestBindingTime=preCompileTime</code></p>
<code>arraySizeHandling</code>	<code>ArraySizeHandlingEnum</code>	0..1	attr	The way how the size of the array is handled in case of a variable size array.
<code>arraySizeSemantics</code>	<code>ArraySizeSemanticsEnum</code>	0..1	attr	This attribute controls the meaning of the value of the array size.







Class	ImplementationDataTypeElement			
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ImplementationDataTypeElement as optional. This means that, at runtime, the ImplementationDataTypeElement may or may not have a valid value and shall therefore be ignored.  The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end.
subElement (ordered)	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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElement.

**Table 5.21: ImplementationDataTypeElement**

### 5.2.5.2 Modeling of Type Reference using Implementation Data Type

[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.](RS\_SWCT\_03217)

[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), these properties overwrite the properties of the refined ImplementationDataType.](RS\_SWCT\_03217)

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.

[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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.

### 5.2.5.3 Modeling of Structure using Implementation Data Type

**[constr\_1106] Structure shall have at least one element** [An `ImplementationDataType` or `ImplementationDataTypeElement` of category `STRUCTURE` shall own at least one `ImplementationDataTypeElement` **at the time when the contract phase generation is executed.**]()

#### 5.2.5.3.1 Modeling of Optional Element Structure with ImplementationDataType

The definition of an `ImplementationDataType` that represents an `Optional Element Structure` shall not only rely on the existence of optional elements.

Also, the definition of the enclosing `ImplementationDataType` shall clearly signal the intention by means of the dedicated attribute `ImplementationDataType.isStructWithOptionalElement`.

**[TPS\_SWCT\_01772] Semantics of attribute `ImplementationDataType.isStructWithOptionalElement`** [If attribute `ImplementationDataType.isStructWithOptionalElement` is set to `True` then the `ImplementationDataType` advertises the intention to represent an `Optional Element Structure` such that the fulfillment of structural requirements for the existence of optional elements can be formally checked.

Again, this attribute represents a formal specification that optionality is intended as opposed to an `ImplementationDataType` that fulfills the structural requirements out of different motivations.](*RS\_SWCT\_03217, RS\_SWCT\_03320*)

**[TPS\_SWCT\_01773] Definition of `Optional Element Structure` on the level of `ImplementationDataType`** [The modeling approach for the definition of an `Optional Element Structure` on the level of `ImplementationDataType` is to set the attribute `ImplementationDataTypeElement.isOptional` to the value `True`.

If the attribute is not set or set to the value `False` then the respective `ImplementationDataTypeElement` shall be considered mandatory.]([RS\\_SWCT\\_03217](#), [RS\\_SWCT\\_03320](#))

**[constr\_1637] Existence of `ImplementationDataTypeElement.isOptional` vs. `ImplementationDataType.isStructWithOptionalElement`** [If one `ImplementationDataType.subElement` sets attribute `isOptional` to the value `True` then the enclosing `ImplementationDataType` shall also set attribute `isStructWithOptionalElement` to `True`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

In order to be able to generate a proper RTE API for the access to optional elements of data types in general it is necessary to impose structural requirements on the definition of `ImplementationDataType`.

In particular, it is necessary at runtime to store the information about the availability of a specific `ImplementationDataTypeElement` where attribute `isOptional` has been set to the value `True` in the context of an `ImplementationDataType` of category `STRUCTURE`.

An `ImplementationDataType` that represents an `Optional Element Structure` shall contain a special element which represents an *availability bitfield*.

This bitfield is implemented as an array of `uint8` and shall hold one bit for each optional element contained in the structured data type.

In particular, the applicable structural requirements for an `ImplementationDataType` that represents an `Optional Element Structure` are described in the following specification items.

**[TPS\_SWCT\_01774] Modeling of `ImplementationDataType` with optional elements** [

The following approach shall be taken to model an `ImplementationDataType` that represents an `Optional Element Structure`:

- The first `ImplementationDataTypeElement` of `ImplementationDataType` where attribute `isStructWithOptionalElement` is set to `True` shall have the `shortName` `availabilityBitfield`. [[constr\\_1638](#)] applies.
- This `ImplementationDataTypeElement` shall be of category `ARRAY`
- The `ImplementationDataTypeElement` shall set attribute `arraySizeSemantics` to the value `fixedSize`.
- The `ImplementationDataTypeElement` shall aggregate a further `ImplementationDataTypeElement` in the role `subElement` for which the following requirements apply:

- The `ImplementationDataTypeElement` shall be of category `TYPE_REFERENCE` that eventually refers to an `ImplementationDataType` that - one way or the other - implements an array of unsigned bytes, e.g. take the `Platform Data Type` named `uint8` as the element type<sup>10</sup>.
- The `ImplementationDataTypeElement` shall set the value of attribute `arraySize` to `max(1,ceil(numberOfOptionalElements / 8))`.

](*RS\_SWCT\_03320*)

**[constr\_1638] First `ImplementationDataTypeElement` of `ImplementationDataType` that represents an `Optional Element Structure`** [The first `ImplementationDataTypeElement` of `ImplementationDataType` that represents an `Optional Element Structure`, i.e. the `availabilityBitfield` according to [TPS\_SWCT\_01774], shall not set attribute `isOptional` to `True` **at the time when the contract phase generation is executed.**]()

A further structural requirement applies.

**[constr\_1639] `ImplementationDataTypeElement` with attribute `isOptional` set to `True`** [`ImplementationDataTypeElement` with attribute `isOptional` set to `True` shall **not** be of category `STRUCTURE` **at the time when the contract phase generation is executed.**]()

Instead, nested structures shall be created by modeling `ImplementationDataTypeElements` of category `TYPE_REFERENCE` that in turn refer to `ImplementationDataTypes` of category `STRUCTURE`.

Rationale: the existence of [constr\_1639] simplifies the concept of the availability bit-field.

The bitfield shall **only** contain information of the availability of the direct child elements and **not** of elements of sub-structures.

By using the category `TYPE_REFERENCE` it is assured that a separate `ImplementationDataType` of category `STRUCTURE` is generated for the sub-structure.

Since the AUTOSAR RTE provides the APIs to access the availability information on the basis of an `ImplementationDataType` of category `STRUCTURE`, the usage of anonymous structures with optional elements is not possible.

#### 5.2.5.4 Modeling of Union using Implementation Data Type

**[constr\_1107] Union shall have at least one element** [An `ImplementationDataType` or `ImplementationDataTypeElement` of category `UNION` shall own at least one `ImplementationDataTypeElement` **at the time when the contract phase generation is executed.**]()

<sup>10</sup>this relation could be expressed in a more formal way. But it would be a very expansive formal way in an already complicated specification item. It is assumed that it is sufficient to convey the general idea.

**[TPS\_SWCT\_01759] Use cases for unions** [There are different use cases for the definition of a union data type:

1. The `DataPrototypes` derived from the union data type shall be transported over a communication network. For this purpose, it is necessary to apply a special modeling in the form of a wrapped union data type, as explained by [\[TPS\\_SWCT\\_01700\]](#).
2. The `DataPrototypes` created from the union data type are used internally within the same ECU, e.g. as a `PerInstanceMemory`, `romBlock`, or `ramBlock`. In this case the modeling of the union data type does not depend on specific constraints.

]([RS\\_SWCT\\_03217](#))

In summary, there are cases where unions can be used in `PortInterfaces`, but these are restricted to the fulfillment of certain conditions that are explained in [\[constr\\_1607\]](#).

**[constr\_1607] Only Wrapped Union Data Types in PortInterface** [Within the scope of a `PortInterface` the usage of a Union data type is only supported

- for `Wrapped Union Data Types`.
- for a `PortInterface` that is used to type a `PortPrototype` that does not appear as a context in an `instanceRef` owned by a `DataMapping`. See also [\[constr\\_1441\]](#).

This rule shall be imposed **at the time when the contract phase generation is executed.**]([\(\)](#))

## 5.2.5.5 Modeling of Array using Implementation Data Type

### 5.2.5.5.1 Overview

**[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.]([RS\\_SWCT\\_03217](#))

**[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`

](RS\_SWCT\_03217)

**[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 unless attribute `ImplementationDataTypeElement.arraySizeHandling` exists and is set to the value `inheritedFromArrayElementTypeSize`.

This rule shall be imposed **at the time when the contract phase generation is executed.**](/)

**[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`).](RS\_SWCT\_03217)

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)

**[constr\_1783] Existence of attribute `ImplementationDataTypeElement.arrayImplPolicy`** [Attribute `ImplementationDataTypeElement.arrayImplPolicy` shall only exist **at the time when the contract phase generation is executed** if the enclosing `ImplementationDataType` or `ImplementationDataTypeElement` is of category `ARRAY`.](/)

The usage of attribute `ImplementationDataTypeElement.arrayImplPolicy` does not have an impact on model semantics, i.e. it does not impose further conditions on the modeling of arrays.

The attribute merely influences the implementation of the respective array in the programming language binding in the form of generated C code.

<b>Enumeration</b>	<code>ArrayImplPolicyEnum</code>
<b>Package</b>	<code>M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes</code>
<b>Note</b>	This meta-class provides values to configure the implementation of the payload part of an array.
<b>Literal</b>	<i>Description</i>





Enumeration	ArrayImplPolicyEnum
payloadAsArray	This configuration demands the implementation of the payload as an array. <b>Tags:</b> atp.EnumerationLiteralIndex=0
payloadAsPointerTo Array	This configuration demands the implementation of the payload as a pointer to an array. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 5.22: ArrayImplPolicyEnum**

### 5.2.5.5.2 Modeling of Variable-Size Array using Implementation Data Type

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 [5.2.4.2.1.1](#).

**[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\\_03217](#), [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\\_03217](#), [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](#).

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\\_03217](#), [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\_03217, RS\_SWCT\_03181)

The statement made by [TPS\_SWCT\_01612] allows the specification of coherency 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\_03217, 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\_03217, 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\_03217, RS\_SWCT\_03181)

For reasons of readability and comprehensibility 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 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 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]

**[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 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 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 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 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.** `()`

**[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 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 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 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 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.** |()

**[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 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 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 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 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 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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` **at the time when the contract phase generation is executed.**]()

**[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 **at the time when the contract phase generation is executed.**]()

**[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\_03217, 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 the 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 `maxNumberOfElements`).

This is the `Size Indicator` which holds the current number of valid elements of the variable size array.](RS\_SWCT\_03217, 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 the 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 `arraySize`).

This is the `Size Indicator` which holds the current number of valid elements of the `Variable-Size Array Data Type`.](RS\_SWCT\_03217, 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\_03217, 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\_03217, 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 `maxNumberOfElements`).](RS\_SWCT\_03217, 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 `maxNumberOfElements` of the corresponding `ApplicationArrayDataType`.](RS\_SWCT\_03217, 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\\_03217](#), [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 [5.43](#) shows which `SwDataDefProps` will be typically used here.

**[TPS\_SWCT\_01257] `ImplementationDataType` or the aggregated `ImplementationDataTypeElements` do not form closed sets** [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](#), [RS\\_SWCT\\_03217](#))

At the end, all the “leaves” of the complete tree formed by these references shall end up in `SwBaseTypes`.

Figures [5.16](#), [5.17](#), and Figure [5.18](#) illustrate more examples about Typedefs and references.

### 5.2.5.6 Modeling of Pointer using Implementation Data Type

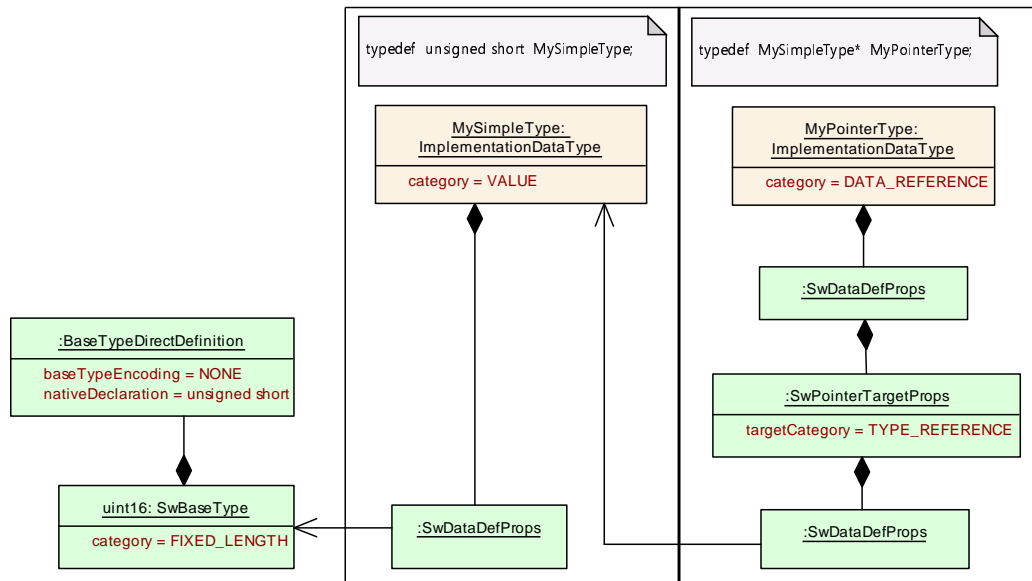


Figure 5.16: Example (1) for TypeDefs

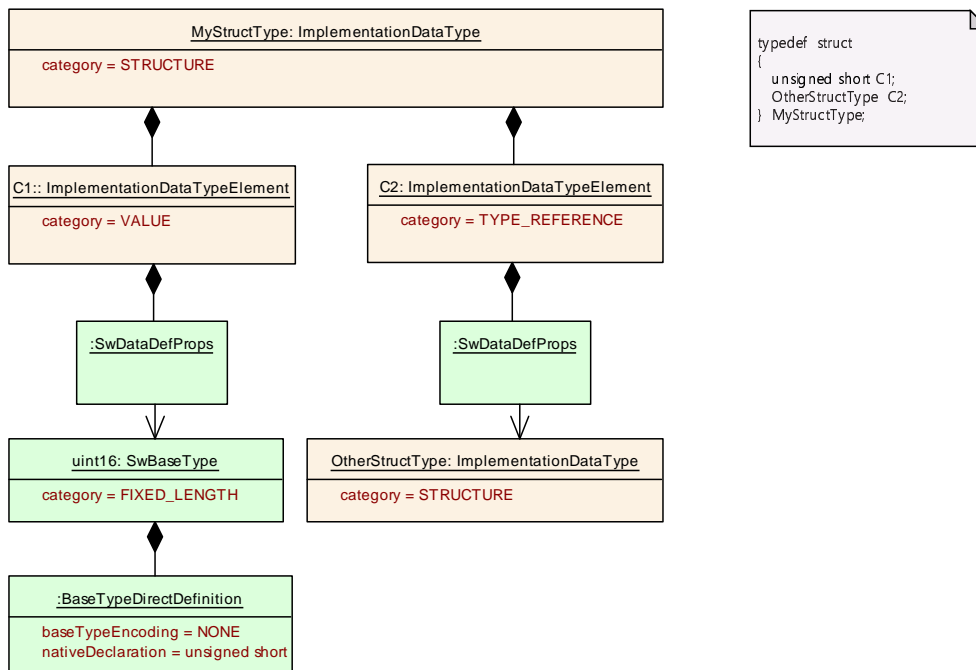
**[TPS\_SWCT\_01258] Definition of a pointer to data** [The definition of a data pointer requires a special meta-class `SwPointerTargetProps` which aggregates another `SwDataDefProps`. This mechanism allows to describe the `category` and properties of the pointer object itself as well as the `category` 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 to this rule applies if the `swDataDefProps` owned by the `SwPointerTargetProps` refers to a `SwBaseType` with native type declaration `void`, in this case `VALUE` is also permitted.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()





**Figure 5.17: 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**()

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 5.18) is the only viable way that is positively supported by the AUTOSAR RTE [2].

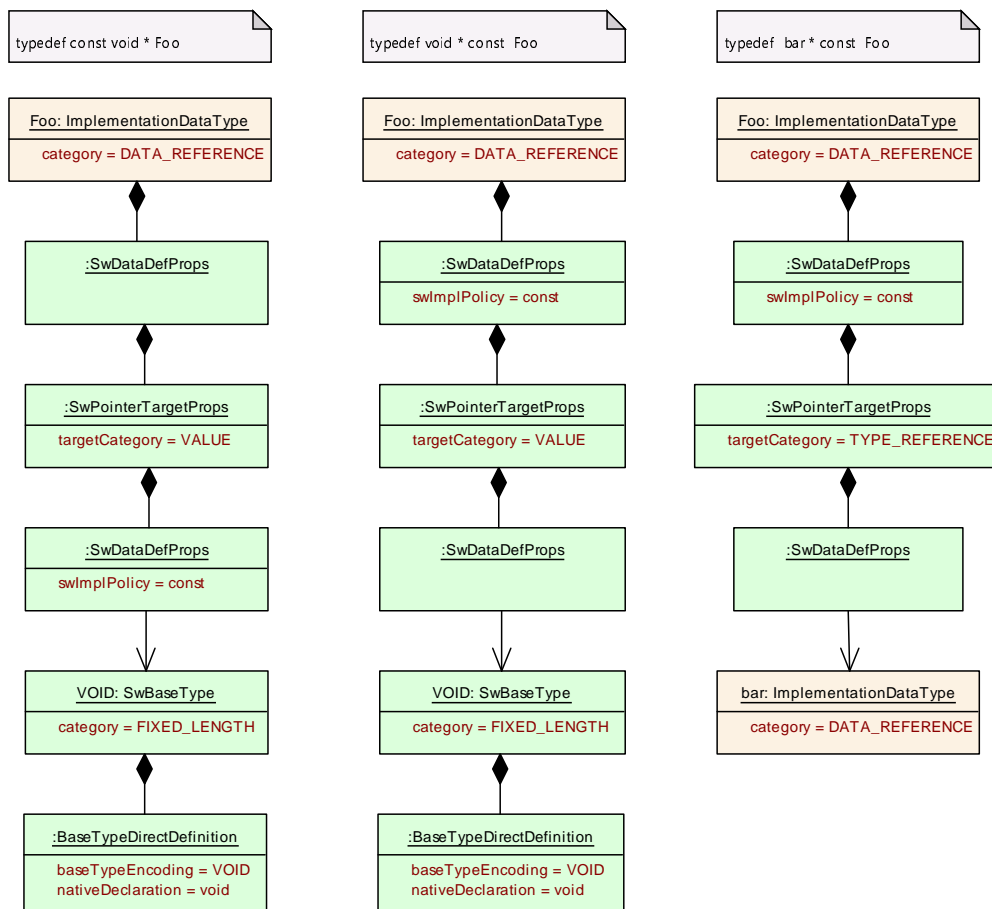


Figure 5.18: Example (3) for TypeDefs

[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 [6].] (*RS\_SWCT\_03217*)

<b>Class</b>	<b>SwPointerTargetProps</b>			
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Tags:</b> xml.sequenceOffset=40
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of the target data type.  <b>Tags:</b> xml.sequenceOffset=30
targetCategory	Identifier	0..1	attr	This specifies the category of the target: <ul style="list-style-type: none"> <li>In case of a data pointer, it shall specify the category of the referenced data.</li> <li>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.</li> </ul> <b>Tags:</b> xml.sequenceOffset=5

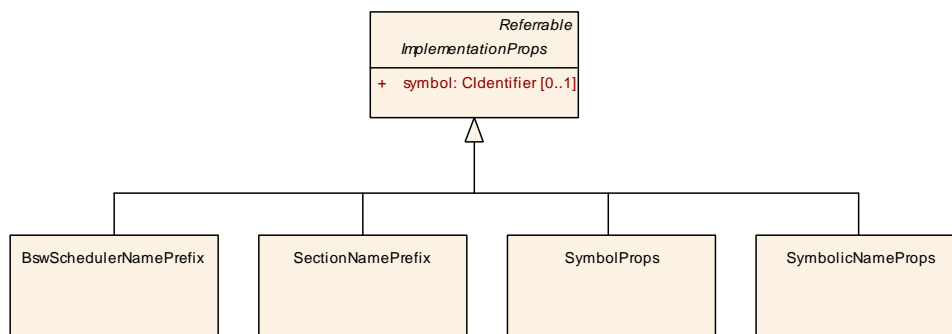
**Table 5.23: SwPointerTargetProps**

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 about [symbolProps](#), please refer to [Figure 5.15](#).



**Figure 5.19: [ImplementationProps](#) and its subclasses**

<b>Class</b>	<b>ImplementationProps</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Implementation			
<b>Note</b>	Defines a symbol to be used as (depending on the concrete case) either a complete replacement or a prefix when generating code artifacts.			
<b>Base</b>	ARObject, Referrable			
<b>Subclasses</b>	BswSchedulerNamePrefix, ExecutableEntityActivationReason, SectionNamePrefix, SymbolProps, SymbolicNameProps			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
symbol	CIdentifier	0..1	attr	The symbol to be used as (depending on the concrete case) either a complete replacement or a prefix.

**Table 5.24: ImplementationProps**

**[constr\_1909] Existence of attribute `ImplementationProps.symbol`** [For each `ImplementationProps`, the attribute `symbol` shall exist **at the time when the contract phase generation is executed.**]()

<b>Class</b>	<b>SymbolProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	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.			
<b>Base</b>	ARObject, ImplementationProps, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 5.25: SymbolProps**

## 5.2.6 Base Type

**[TPS\_SWCT\_01260] `SwBaseType`** [`BaseType` is used to specify the basic data type level. AUTOSAR uses 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 typedefs 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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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 symbol<sup>11</sup> 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.

**[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 [23].

**[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 [6] for details on these use cases.] ()

A more detailed description of `BaseTypes` can also be found in *ASAM MCD 2 Harmonized Data Objects*.<sup>12</sup>

<b>Class</b>	<b>BaseType</b> (abstract)			
<b>Package</b>	M2::MSR::AsamHdo::BaseTypes			
<b>Note</b>	This abstract meta-class represents the ability to specify a platform dependent base type.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">SwBaseType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>



<sup>11</sup>Note that the 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.

<sup>12</sup>The definition of *Harmonized Data Objects* can be retrieved from ASAM at [www.asam.net](http://www.asam.net). Access is limited to ASAM members.



Class	BaseType (abstract)			
baseType Definition	<a href="#">BaseTypeDefinition</a>	1	aggr	This is the actual definition of the base type. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

**Table 5.26: BaseType**

The aggregation [BaseType.baseTypeDefinition](#) affects two abstract meta-classes and is therefore never used directly. However, in order to not break the semantics of the derived meta-classes [SwBaseType](#) and [BaseTypeDirectDefinition](#) a constraint is still required.

**[constr\_1910] Existence of attribute [BaseType.baseTypeDefinition](#)** [For each [BaseType](#) (which will be utilized in the form of [SwBaseType](#)), the aggregation in the role [baseTypeDefinition](#) shall exist **at the time when the contract phase generation is executed.**] ()

Class	SwBaseType			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This meta-class represents a base type used within ECU software. <b>Tags:</b> atp.recommendedPackage=BaseTypes			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">BaseType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

**Table 5.27: SwBaseType**

Class	BaseTypeDefinition (abstract)			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This meta-class represents the ability to define a basetype.			
Base	<a href="#">ARObject</a>			
Subclasses	<a href="#">BaseTypeDirectDefinition</a>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

**Table 5.28: BaseTypeDefinition**

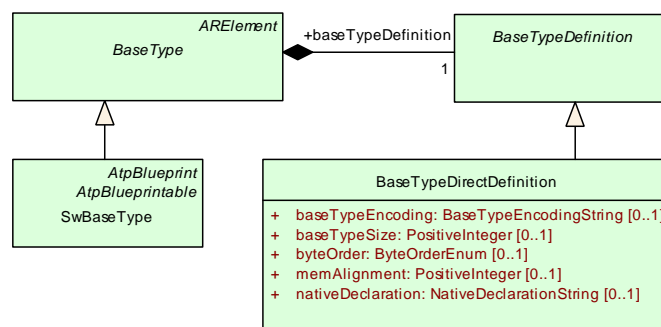
Class	BaseTypeDirectDefinition			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	<a href="#">ARObject</a> , <a href="#">BaseTypeDefinition</a>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–





Class	BaseTypeDirectDefinition			
baseTypeEncoding	BaseTypeEncodingString	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. <b>Tags:</b> xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. <b>Tags:</b> xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. <b>Tags:</b> xml.sequenceOffset=110
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". <b>Tags:</b> xml.sequenceOffset=100
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 shortName: "MyUnsignedInt" native Declaration: "unsigned short"  Results in  typedef unsigned short MyUnsignedInt;  If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE.  If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size.  This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems.  <b>Tags:</b> xml.sequenceOffset=120

**Table 5.29: BaseTypeDirectDefinition**



**Figure 5.20: Base Type**

Some additional hints to the properties of `SwBaseType`:

- **[constr\_1011] category of SwBaseType** [For the attribute `SwBaseType.category` only the values `FIXED_LENGTH` and `VOID` are supported **at the time when the contract phase generation is executed.**]()

- **[constr\_1422] Value of `category` is `VOID`** [If the value of the attribute `SwBaseType.category` is set to `VOID` then the attribute `baseTypeSize` and `baseTypeEncoding` shall not exist **at the time when the contract phase generation is executed.**]()
- **[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 **at the time when the contract phase generation is executed.**]()
- **[TPS\_SWCT\_01444] Size of `SwBaseType` is specified in bits** [In both cases (mentioned in [constr\_1012]) 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: single-byte coded character
- ISO-8859-2: single-byte coded character
- WINDOWS-1252: single-byte coded character
- UTF-8: UCS Transformation Format 8
- UTF-16: Character encoding for Unicode *code points* based on 16 bit *code units* [15]
- 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`.

This rule shall be imposed **at any time in the workflow.**]()

- **[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 [6]) 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

There is already predefined terminology (see [15]) 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].

<b>Enumeration</b>	<b>ByteOrderEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
<b>Note</b>	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.
<b>Literal</b>	<b>Description</b>
mostSignificantByteFirst	Most significant byte shall come at the lowest address (also known as BigEndian or as Motorola-Format) <b>Tags:</b> atp.EnumerationLiteralIndex=0





Enumeration	ByteOrderEnum
mostSignificantByteLast	Most significant byte shall come highest address (also known as LittleEndian or as Intel-Format) <b>Tags:</b> atp.EnumerationLiteralIndex=1
opaque	For opaque data endianness conversion has to be configured to Opaque. See AUTOSAR COM Specification for more details. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 5.30: ByteOrderEnum**

**[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 [15].]()

**[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 [15].]()

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 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

]([RS\\_SWCT\\_03216](#), [RS\\_SWCT\\_03217](#))

**[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].

]([RS\\_SWCT\\_03216](#), [RS\\_SWCT\\_03217](#))

### 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 `ApplicationPrimitiveDataType` of category `STRING`.] (*RS\_SWCT\_03216*)

**[constr\_1241] Compound Primitive Data Types and invalidValue** [Compound Primitive Data Types that have set the value of `category` other than `STRING` shall **not** define `invalidValue` **at the time when the contract phase generation is executed**.] ()

### 5.2.7.3 Integral Primitive Type

The `SenderReceiverToSignalMapping` (see [10]) allows for the integral mapping of a piece of data to a single `SystemSignal`. The specification of AUTOSAR COM [22] 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 `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` 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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1230] ApplicationDataType that qualifies for Integral Primitive Type** [An `ApplicationDataType` qualifies as an `Integral Primitive Type` if and only if **all** 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]

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

#### 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 2.7.

#### 5.2.7.5 Wrapped Union Data Type

There are use cases for sending a `DataPrototype` that is effectively typed by a **union** data type over a communication network. In this case, however, it is necessary to not only send the `DataPrototype` itself but add an information about the applicable member of the union as a form of “meta-data” to the transmission.

By this means the sender can identify the applicable member of the union and the receiver can accordingly access the proper union element.

It is the nature of union data types that executable code shall **symmetrically** access the union, i.e. the member that was written needs to be read, the usage of a union as a “type converter” is heavily frowned upon (because it causes unspecified behavior from ISO-C:99 [24] point of view) and shall be discouraged by AUTOSAR.

Thus, AUTOSAR needs to take this condition into account and define a specific modeling for the handling of union data types.

**[TPS\_SWCT\_01700] Definition of unions that can be transmitted over a communication network** [If it is intended to send a data object typed by a union data type over a communication bus then a specific modeling is required for this purpose.

- The union data type shall never be used as such, it shall always be enclosed in an `ImplementationDataType` of category `STRUCTURE` that aggregates exactly two `ImplementationDataTypeElements`:
  - The **first** `ImplementationDataTypeElement` shall be used to identify the applicable element of the actual union data type.

The `shortName` of this element shall be set to **“memberSelector”**, it shall be of category `VALUE`, or of category `TYPE_REFERENCE` that finally boils down to category `VALUE`.

Furthermore, it shall refer to a `SwBaseType` with attribute `baseTypeEncoding` set to `NONE` and attribute `baseTypeSize` set to the value 8, 16, or 32.

This `ImplementationDataTypeElement` shall be called the `Member Selector`.

- The **second** `ImplementationDataTypeElement` shall be of category `UNION`, it represents the actual union “payload”.
- The purpose of the `Member Selector` is to identify the element of the union data type that applies for a given access to the union.

If the value of the `Member Selector` is set to 1 then the first `subElement` of the `ImplementationDataType` of category `UNION` is applicable.

If the value of the `Member Selector` is set to 2 then the second `subElement` is applicable and so on.

- The value of the `Member Selector` shall range between **the value 1** and the number of `subElements` of the `ImplementationDataTypeElement` of category `UNION`. Once again, the index counting is 1-based!
- Obviously, the actual data type used to hold the `Member Selector` shall be capable of storing a value that corresponds to the number of `subElements` of the `ImplementationDataTypeElement` of category `UNION`.
- Constraint [constr\_1441] applies.

]([RS\\_SWCT\\_03217](#))

**[TPS\_SWCT\_01701] Wrapped Union Data Type** [Data types that fulfill the requirements of [TPS\_SWCT\_01700] shall be called `Wrapped Union Data Types`.]([RS\\_SWCT\\_03217](#))

**[constr\_1442] category `TYPE_REFERENCE` shall not be used for modeling the “payload” of a `Wrapped Union Data Type`** [For the modeling of the “payload” part of a `Wrapped Union Data Type` it shall not be possible **at any time in the**

**workflow** to use an `ImplementationDataElement` of category `TYPE_REFERENCE` that finally (i.e. after all possible indirections are resolved) boils down to category `UNION`.]()

The definition of the `Wrapped Union Data Type` represents the **canonical way** of how union data types shall be used in AUTOSAR on the application and communication level. Consequentially, the usage of the `category` value `UNION` is effectively limited to an `ImplementationDataElement`.

**[constr\_1444] Limited applicability of Wrapped Union Data Type** [There is no support **at any time in the workflow** for the usage of `Wrapped Union Data Type` in `PortInterfaceMappings`, and `Diagnostics`.]()

For the time being, AUTOSAR restricts the initialization of a union data type to the first member of the union data type, see [\[constr\\_1445\]](#).

#### Listing 5.9: Simplified example of a `Wrapped Union Data Type`

```

<IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>UnionExample</SHORT-NAME>
  <CATEGORY>STRUCTURE</CATEGORY>
  <SUB-ELEMENTS>
    <IMPLEMENTATION-DATA-TYPE-ELEMENT>
      <SHORT-NAME>memberSelector</SHORT-NAME>
      <CATEGORY>VALUE</CATEGORY>
    </IMPLEMENTATION-DATA-TYPE-ELEMENT>
    <IMPLEMENTATION-DATA-TYPE-ELEMENT>
      <SHORT-NAME>payload</SHORT-NAME>
      <CATEGORY>UNION</CATEGORY>
      <SUB-ELEMENTS>
        <IMPLEMENTATION-DATA-TYPE-ELEMENT>
          <SHORT-NAME>primitive</SHORT-NAME>
          <CATEGORY>VALUE</CATEGORY>
        </IMPLEMENTATION-DATA-TYPE-ELEMENT>
        <IMPLEMENTATION-DATA-TYPE-ELEMENT>
          <SHORT-NAME>array</SHORT-NAME>
          <CATEGORY>ARRAY</CATEGORY>
          <SUB-ELEMENTS>
            <IMPLEMENTATION-DATA-TYPE-ELEMENT>
              <SHORT-NAME>arraySub</SHORT-NAME>
              <CATEGORY>VALUE</CATEGORY>
              <ARRAY-SIZE>4</ARRAY-SIZE>
              <ARRAY-SIZE-SEMANTICS>FIXED-SIZE</ARRAY-SIZE-SEMANTICS>
            </IMPLEMENTATION-DATA-TYPE-ELEMENT>
          </SUB-ELEMENTS>
        </IMPLEMENTATION-DATA-TYPE-ELEMENT>
      </SUB-ELEMENTS>
    </IMPLEMENTATION-DATA-TYPE-ELEMENT>
  </SUB-ELEMENTS>
</IMPLEMENTATION-DATA-TYPE>

```

One obvious consequence of this restriction is that for any given `ValueSpecification` taken to initialize a `Wrapped Union Data Type` the value of the `Member Selector` is **strictly** locked to 1.

**[constr\_1445] Initialization of the Member Selector of a Wrapped Union Data Type** [The `initValue` for the `Member Selector` shall never be set to any value other than 1.

This rule shall be imposed **at any time in the workflow.**]()

Another aspect of the initialization of a `Wrapped Union Data Type` is that the “payload” part cannot be treated as a composite data type unless the first element of the “payload” part is typed by a composite data type.

In other words, it is not possible to initialize the first `subElement` of an `ImplementationDataTypeElement` of category `UNION`. It is only possible to assign an initial value to the “payload” part itself.

**[TPS\_SWCT\_01702] Initialization of the “payload” of a Wrapped Union Data Type** [The `initValue` for the `ImplementationDataTypeElement` of category `UNION` shall be assigned to the `ImplementationDataTypeElement` of category `UNION` but it shall reflect the structure of the first `subElement` of the `ImplementationDataTypeElement` of category `UNION`.] (*RS\_SWCT\_03217*)

In other words, if the first `subElement` of the `ImplementationDataTypeElement` of category `UNION` is of a primitive type then a `NumericalValueSpecification` shall be used to initialize the `ImplementationDataTypeElement` of category `UNION`.

If the `subElement` is typed by a composite data type then a `CompositeValueSpecification` shall be used to initialize the `ImplementationDataTypeElement` of category `UNION`.

To summarize the initialization issue, a `Wrapped Union Data Type` is modeled as a structure of two elements and requires a `RecordValueSpecification` that in turn aggregates two `ValueSpecifications`, one for the `Member Selector` that shall have no other value than 1, and one for the “payload”.

The structure of the second `ValueSpecification` depends on the data type used for the first element of the “payload”.

The following example shows a simplified and stripped-down (e.g. without the `Sw-DataDefProps` required to make the model complete) model of a `Wrapped Union Data Type`.

### 5.2.7.6 Optional Element Structure

As already mentioned in section 2.8, there are use cases for structured data types that contain optional elements that may or may not exist at a given time.

These data types require a specific modeling on both the level of `Application-DataType` and the level of `ImplementationDataType`.



**[TPS\_SWCT\_01775] Structured data types with optional elements** [A structured data type that contains at least one optional element shall be called an `Optional Element Structure`.] ([RS\\_SWCT\\_03320](#))

On the level of `ApplicationDataType`, the existence of optional elements is signaled by setting the attribute `ApplicationRecordElement.isOptional` to `True`. For more details, please refer to section [5.2.4.2.2](#).

The description of how an `Optional Element Structure` shall be modeled using `ImplementationDataType` can be found in section [5.2.5.3.1](#).

## 5.3 Data Prototypes

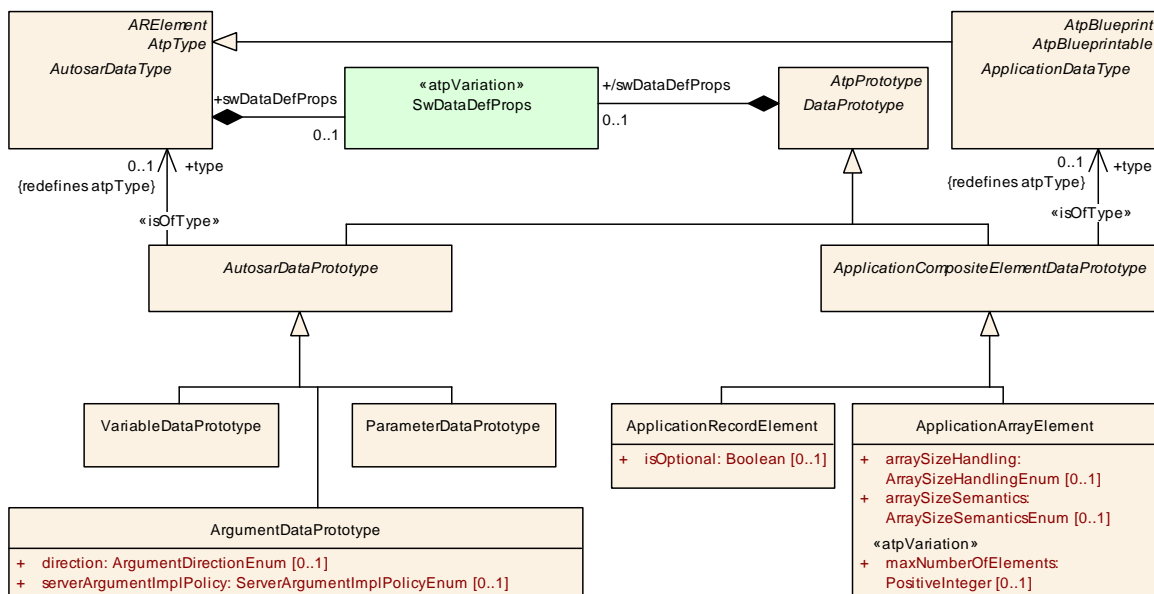
### 5.3.1 Overview

**[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 [5.21](#).

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 5.21: Data Prototypes Overview**

<b>Class</b>	<b>DataPrototype</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	Base class for prototypical roles of any data type.			
<b>Base</b>	<i>ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Subclasses</b>	<i>ApplicationCompositeElementDataPrototype, AutosarDataPrototype</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swDataDef Props	<a href="#">SwDataDefProps</a>	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level.

**Table 5.31: DataPrototype**

<b>Class</b>	<b>AutosarDataPrototype</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	Base class for prototypical roles of an AutosarDataType.			
<b>Base</b>	<i>ARObject, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Subclasses</b>	<i>ArgumentDataPrototype, ParameterDataPrototype, VariableDataPrototype</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
type	<a href="#">AutosarDataType</a>	0..1	tref	This represents the corresponding data type. <b>Stereotypes:</b> isOfType

**Table 5.32: AutosarDataPrototype**

<b>Class</b>	<b>ApplicationCompositeElementDataPrototype</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	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 instance Refs.			
<b>Base</b>	<i>ARObject, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Subclasses</b>	<i>ApplicationArrayElement, ApplicationRecordElement</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
type	<a href="#">ApplicationDataType</a>	0..1	tref	This represents the corresponding data type. <b>Stereotypes:</b> isOfType

**Table 5.33: 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 SwDataDef-Props** [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 about this aspect of the meta-model.

**[TPS\_SWCT\_01445] Applicability of SwDataDefProps for DataPrototypes** [The applicability of SwDataDefProps for DataPrototypes shall follow the same rules as for the categories of the corresponding AutosarDataTypes.]()

The applicability of SwDataDefProps for DataPrototypes is documented in Table 5.7.

Further information can be found in table 5.34 and table 5.35.

Please note that table 5.34 does not include the ApplicationRecordElement and ApplicationArrayElement because these specializations of ApplicationCompositeElementDataPrototype are already part of table 5.7. The same applies for table 5.35 which does not include the ImplementationDataTypeElement.

Attributes of SwDataDefProps	Root EI.			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	
additionalNativeTypeQualifier																	
annotation	x	x	x	*	*	*	*	*	*	*	*	*	*	*	*	*	*
baseType																	
compuMethod																	
dataConstr.dataConstrRule.physConstrs	x	x		0..1	0..1		0..1		0..1			0..1	0..1	0..1	0..1	0..1	0..1
dataConstr.dataConstrRule.internalConstrs	x	x		d/c <sup>1</sup>	d/c		d/c		d/c			d/c	d/c	d/c	d/c	d/c	d/c
displayFormat	x	x		0..1	0..1		0..1	0..1	0..1			0..1	0..1	0..1	0..1	0..1	0..1
displayPresentation	x	x		0..1	0..1		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	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	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	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	0..1
swCalprmAxisSet																	
swCalprmAxisSet.swCalprmAxis/SwAxisGrouped.swCalprmRef		x	x				0..1					0..1	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	0..1	0..1



<sup>13</sup>don't care



Attributes of SwDataDefProps	Root EI.			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
swCalprmAxisSet.swCalprmAxis/SwAxis-Grouped.sharedAxisType																
swCalprmAxisSet.swCalprmAxis/SwAxis-Individual.inputVariableType																
swCalprmAxisSet.swCalprmAxis/SwAxis-Individual.unit																
swComparisonVariable			x									0..1	0..1	0..1	0..1	0..1
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																
swValueBlockSizeMult																
unit																
valueAxisDataType																

Table 5.34: 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 5.34.

This rule shall be imposed **at any time in the workflow.**]()

**[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 5.35.

This rule shall be imposed **at any time in the workflow.**]()

Attributes of SwDataDefProps	Root Element			Attribute Existence per Category						
	DataPrototype	InstantiationDataDefProps	ParameterAccess	VALUE	DATA_REFERENCE	FUNCTION_REFERENCE	TYPE_REFERENCE	STRUCTURE	UNION	ARRAY
additionalNativeTypeQualifier										
annotation	X	X	*	*	*	*	*	*	*	*
baseType										
compuMethod										
dataConstr.dataConstrRule.physConstrs	X	X		d/c <sup>14</sup>			d/c			d/c
dataConstr.dataConstrRule.internalConstrs	X	X		0..1			0..1			0..1
displayFormat	X	X		0..1			0..1	0..1	0..1	0..1
displayPresentation	X	X		0..1			0..1			0..1
implementationDataType										
invalidValue										
stepSize	X	X		0..1						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										
swValueBlockSizeMult										
unit										
valueAxisDataType										

Table 5.35: Allowed Attributes vs. category for DataPrototypes typed by ImplementationDataTypes

<sup>14</sup>don't care

[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.

]()

<b>Class</b>	<b>VariableDataPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	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.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">AutosarDataPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initValue	<a href="#">ValueSpecification</a>	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

**Table 5.36: VariableDataPrototype**

<b>Class</b>	<b>ParameterDataPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	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.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">AutosarDataPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initValue	<a href="#">ValueSpecification</a>	0..1	aggr	Specifies initial value(s) of the ParameterDataPrototype

**Table 5.37: 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 5.22.

[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.]()

For more information, please refer to chapter 4.5.

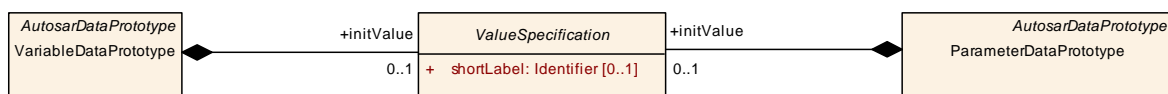


Figure 5.22: Initial value for `AutosarDataPrototypes`

Find more information about the interpretation of `initValue` in section 5.7.

[constr\_1416] Existence of `ApplicationArrayElement.maxNumberOfElements` [The attribute `ApplicationArrayElement.maxNumberOfElements` shall exist at the time when the contract phase generation is executed for all `ApplicationArrayElements` defined in the scope of an `ApplicationArrayDataType` where the attribute `ApplicationArrayDataType.dynamicArraySizeProfile` does not exist.]()

This means that for fixed-size array data types the attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined for every dimension of the fixed-size array data.

### 5.3.2 Data Constraints for DataPrototypes typed by Array DataTypes

There are cases where it should be possible to reference different `DataConstrs` from `DataPrototypes` of category `ARRAY` typed by either an `ApplicationArrayDataType` or an `ImplementationDataType` of category `ARRAY`.

For example, consider a predefined `AutosarDataType` of category `ARRAY` with `uint8` elements, but without the definition of `DataConstrs`.

This `AutosarDataType` is instantiated several times by means of the existence of `DataPrototypes` of category `ARRAY`.

Each of the derived `DataPrototypes` of category `ARRAY` may have individual upper and lower limits which apply to all elements, e.g. one `DataPrototype` has a range of `0..3` while another has a range of `0..7` for all elements.

This use case occurs, for example, when communicating array-data with different ranges over the network.

The motivation to specify the `DataConstr` at the `DataPrototype` of category `ARRAY` is:

- Make explicit that all derived `DataPrototypes` are based on the same `AutosarDataType`.
- Reduce the number of `AutosarDataTypes` in the model and in the code.
- Specify the `DataConstrs` only once where needed.
- Make explicit that all array elements have the same `DataConstrs`.

The same motivation also applies to the `SwDataDefProps.displayFormat` and `SwDataDefProps.stepSize` attributes of `DataPrototypes` of category `ARRAY`.

AUTOSAR supports this use case under the following conditions:

**[constr\_1407] Definition of `SwDataDefProps.dataConstr` depending on the capabilities of the data type** [The definition of a `SwDataDefProps.dataConstr` according to [constr\_1288] and [constr\_1289] is only supported for a `DataPrototype` of category `ARRAY` if the corresponding `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY` also supports the specification of a `SwDataDefProps.dataConstr`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

The prioritization of `SwDataDefProps.dataConstr` for a `DataPrototype` of category `ARRAY` follows the spirit given in [constr\_1015] for the inheritance of `DataPrototypes` from `AutosarDataTypes`.

**[TPS\_SWCT\_01796] Prioritization of `SwDataDefProps.dataConstr` for a `DataPrototype` of category `ARRAY`** [A `SwDataDefProps.dataConstr` specified for a `DataPrototype` of category `ARRAY` refines the `SwDataDefProps.dataConstr` specified at the corresponding `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY`.]()

**[constr\_1408] Definition of `SwDataDefProps.displayFormat` depending on the capabilities of the data type** [The definition of a `SwDataDefProps.displayFormat` according to [constr\_1288] and [constr\_1289] is only supported for a `DataPrototype` of category `ARRAY` if the corresponding `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY` also supports the specification of a `SwDataDefProps.displayFormat`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

The prioritization of `SwDataDefProps.displayFormat` for a `DataPrototype` of category `ARRAY` follows the spirit given in [constr\_1015] for the inheritance of `DataPrototypes` from `AutosarDataTypes`.



**[TPS\_SWCT\_01797] Prioritization of `SwDataDefProps.displayFormat` for a `DataPrototype` of category `ARRAY`** [A `SwDataDefProps.displayFormat` specified for a `DataPrototype` of category `ARRAY` refines the `SwDataDefProps.displayFormat` specified at the corresponding `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY`.]()

**[constr\_1413] Definition of `SwDataDefProps.stepSize` depending on the capabilities of the data type** [The definition of a `SwDataDefProps.stepSize` according to [constr\_1288] and [constr\_1289] is only supported for a `DataPrototype` of category `ARRAY` if the corresponding `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY` also supports the specification of a `SwDataDefProps.stepSize`.

This rule shall be imposed **at the time when the contract phase generation is executed.**()

The prioritization of `SwDataDefProps.stepSize` for a `DataPrototype` of category `ARRAY` follows the spirit given in [constr\_1015] for the inheritance of `DataPrototypes` from `AutosarDataTypes`.

**[TPS\_SWCT\_01798] Prioritization of `SwDataDefProps.stepSize` for a `DataPrototype` of category `ARRAY`** [A `SwDataDefProps.stepSize` specified for a `DataPrototype` of category `ARRAY` refines the `SwDataDefProps.stepSize` specified at the corresponding `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY`.]()

**[constr\_1409] Definition of `SwDataDefProps.dataConstr` depending on the capabilities of the element data type** [The definition of a `SwDataDefProps.dataConstr` according to [constr\_1007] and [constr\_1009] is only supported for an `ApplicationArrayDataType` or an `ImplementationDataType` of category `ARRAY` if the aggregated `ApplicationArrayDataType.element` or `ImplementationDataType.subElement` also supports the specification of a `SwDataDefProps.dataConstr`.

This rule shall be imposed **at the time when the contract phase generation is executed.**()

**[constr\_1718] Inheritance of `SwDataDefProps.dataConstr` from an array data type to the array elements** [A `SwDataDefProps.dataConstr` specified for an `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY` applies to all array leaf elements represented by (potentially multiple levels of) `ApplicationArrayDataType.element` or `ImplementationDataType.subElement`.

In this case, the `ApplicationArrayDataType.element` or `ImplementationDataType.subElement` shall not have an own `SwDataDefProps.dataConstr`. This also applies for multi-dimensional array data types.

This rule shall be imposed **at the time when the contract phase generation is executed.**()

**[constr\_1410] Definition of `SwDataDefProps.displayFormat` depending on the capabilities of the element data type** [The definition of a `SwDataDefProps.displayFormat` according to [constr\_1007] and [constr\_1009] is only supported for an `ApplicationArrayDataType` or an `ImplementationDataType` of category `ARRAY` if the aggregated `ApplicationArrayDataType.element` or `ImplementationDataType.subElement` also supports the specification of a `SwDataDefProps.displayFormat`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1719] Inheritance of `SwDataDefProps.displayFormat` from an array data type to the array elements** [A `SwDataDefProps.displayFormat` specified for an `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY` applies to all array leaf elements represented by (potentially multiple levels of) `ApplicationArrayDataType.element` or `ImplementationDataType.subElement`.

In this case, the `ApplicationArrayDataType.element` or `ImplementationDataType.subElement` shall not have an own `SwDataDefProps.displayFormat`. This also applies for multi-dimensional array data types.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1414] Definition of `SwDataDefProps.stepSize` depending on the capabilities of the element data type** [The definition of a `SwDataDefProps.stepSize` according to [constr\_1007] and [constr\_1009] is only supported for an `ApplicationArrayDataType` or an `ImplementationDataType` of category `ARRAY` if the aggregated `ApplicationArrayDataType.element` or `ImplementationDataType.subElement` also supports the specification of a `SwDataDefProps.stepSize`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1720] Inheritance of `SwDataDefProps.stepSize` from an array data type to the array elements** [A `SwDataDefProps.stepSize` specified for an `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY` applies to all array leaf elements represented by (potentially multiple levels of) `ApplicationArrayDataType.element` or `ImplementationDataType.subElement`.

In this case, the `ApplicationArrayDataType.element` or `ImplementationDataType.subElement` shall not have an own `SwDataDefProps.stepSize`. This also applies for multi-dimensional array data types.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

### 5.3.3 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`.]()

#### 5.3.3.1 AUTOSAR Variable Ref

**[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`.

]()

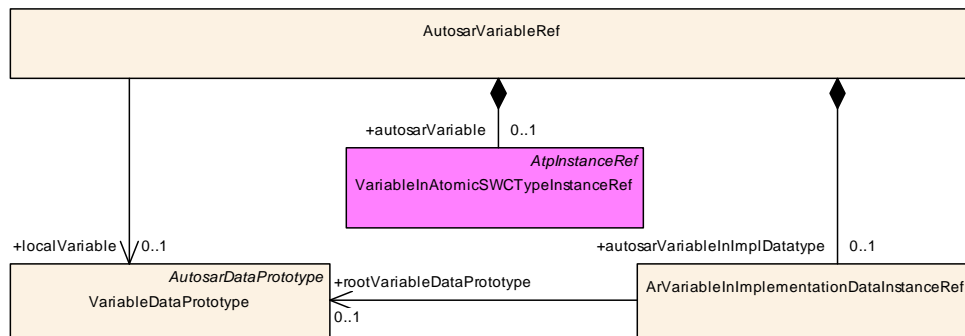
<b>Class</b>	<b>AutosarVariableRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SwComponentTemplate::SwcInternalBehavior::DataElements			
<b>Note</b>	<p>This class represents a reference to a variable within AUTOSAR which can be one of the following use cases:</p> <p>localVariable:</p> <ul style="list-style-type: none"> <li>• localVariable which is used as whole (e.g. InterRunnableVariable, inputValue for curve)</li> </ul> <p>autosarVariable:</p> <ul style="list-style-type: none"> <li>• a variable provided via Port which is used as whole (e.g. dataAccesspoints)</li> <li>• an element inside of a composite local variable typed by ApplicationDatatype (e.g. inputValue for a curve)</li> <li>• an element inside of a composite variable provided via Port and typed by ApplicationDatatype (e.g. inputValue for a curve)</li> </ul> <p>autosarVariableInImplDatatype:</p> <ul style="list-style-type: none"> <li>• an element inside of a composite local variable typed by ImplementationDatatype (e.g. nvram Data mapping)</li> <li>• an element inside of a composite variable provided via Port and typed by Implementation Datatype (e.g. inputValue for a curve)</li> </ul>			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	AutosarVariableRef			
autosarVariable	DataPrototype	0..1	iref	This references a variable which is provided by a port and/or which is part of a CompositeDataType. <b>InstanceRef implemented by:</b> VariableInAtomicSWCTypeInstanceRef
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 5.38: AutosarVariableRef**



**Figure 5.23: Implementation of AutosarVariableRef**

Rules for the modeling and semantics of an `AtpInstanceRef` are defined in [11].

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

### 5.3.3.2 AUTOSAR Parameter Ref

**[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](#) **at the time when the contract phase generation is executed.**]()

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.

<b>Class</b>	<b>AutosarParameterRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
<b>Note</b>	<p>This class represents a reference to a parameter within AUTOSAR which can be one of the following use cases:</p> <p>localParameter:</p> <ul style="list-style-type: none"> <li>localParameter which is used as whole (e.g. sharedAxis for curve)</li> </ul> <p>autosarVariable:</p> <ul style="list-style-type: none"> <li>a parameter provided via PortPrototype which is used as whole (e.g. parameterAccess)</li> <li>an element inside of a composite local parameter typed by ApplicationDatatype (e.g. sharedAxis for a curve)</li> <li>an element inside of a composite parameter provided via Port and typed by ApplicationDatatype (e.g. sharedAxis for a curve)</li> </ul> <p>autosarParameterInImplDatatype:</p> <ul style="list-style-type: none"> <li>an element inside of a composite local parameter typed by ImplementationDatatype</li> <li>an element inside of a composite parameter provided via PortPrototype and typed by ImplementationDatatype</li> </ul>			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
autosarParameter	<a href="#">DataPrototype</a>	0..1	iref	<p>This instance reference is used if the calibration parameter is either imported via a port or is part of a composite data structure.</p> <p><b>InstanceRef implemented by:</b><a href="#">ParameterInAtomicSWCTypeInstanceRef</a></p>
localParameter	<a href="#">DataPrototype</a>	0..1	ref	<p>In the majority of cases this reference goes to <a href="#">ParameterDataPrototypes</a> rather than <a href="#">VariableDataPrototypes</a>. Pointing the reference to a <a href="#">VariableDataPrototype</a> is limited to special use cases, e.g. if the <a href="#">AutosarParameterRef</a> is used in the context of an <a href="#">SwAxisGrouped</a>.</p> <p>This reference is used if the <a href="#">arParameter</a> is local to the current component.</p> <p>Of course, it would technically also be feasible to use an <a href="#">InstanceRef</a> for this case. However, the <a href="#">InstanceRef</a></p>





Class	AutosarParameterRef			
				<p style="text-align: center;">△</p> <p>would not have a contextElement (because the current 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.</p>

**Table 5.39: 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

### 5.3.3.3 Modeling Approach

The attribute `Ref.index` shall be used whenever a model element that represents an array is referenced in a scalar context, i.e. when the reference is supposed to identify a specific array element.

Primitive	Ref			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	<p>This primitive denotes a name based reference. For detailed syntax see the <code>xsd.pattern</code>.</p> <ul style="list-style-type: none"> <li>• first slash (relative or absolute reference) [optional]</li> <li>• Identifier [required]</li> <li>• a sequence of slashes and Identifiers [optional]</li> </ul> <p>This primitive is used by the meta-model tools to create the references.</p> <p><b>Tags:</b>  <code>xml.xsd.customType=REF</code>  <code>xml.xsd.pattern=/?[a-zA-Z][a-zA-Z0-9_]{0,127}/([a-zA-Z][a-zA-Z0-9_]{0,127})*</code>  <code>xml.xsd.type=string</code></p>			
Attribute	Type	Mult.	Kind	Note
base	Identifier	0..1	attr	<p>This attribute reflects the base to be used for this reference.</p> <p><b>Tags:</b><code>xml.attribute=true</code></p>
blueprintValue	String	0..1	attr	<p>This represents a description that documents how the value shall be defined when deriving objects from the blueprint.</p> <p><b>Tags:</b>  <code>atp.Status=draft</code>  <code>xml.attribute=true</code></p>





Primitive	Ref			
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. <b>Tags:</b> xml.attribute=true

**Table 5.40: Ref**

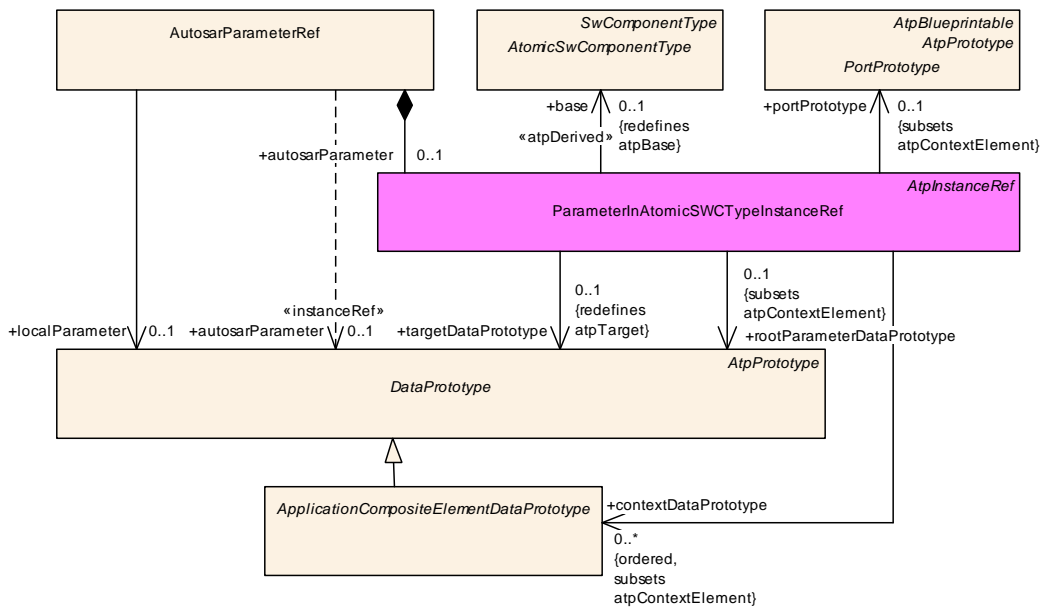
A very typical example for such a situation is the access to an element of an `ApplicationArrayDataType` by means of a reference to `ApplicationArrayElement` in which the index attribute is used.

The usage of the `index` attribute does not make sense if the context of the access is already scalar, e.g. accessing an `ApplicationRecordElement` in the context of an `ApplicationRecordDataType`, the usage of the attribute `index` in a context that is already scalar by nature would also be misleading and is therefore prohibited.

**[constr\_1161] Applicability of the attribute `Ref.index`** [The usage of attribute `Ref.index` is limited to references to the following meta-classes:

- `ApplicationArrayElement`
- Sub-classes of `AbstractImplementationDataTypeElement`.

This rule shall be imposed **at any time in the workflow.**]()



**Figure 5.24: Implementation of the InstanceRef for `AutosarParameterRef`**

Note: If the target of the `AtpInstanceRef` is an `AutosarDataPrototype` then the `rootVariableDataPrototype` shall not exist.

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` for the case that the `targetDataPrototype` is an `ApplicationCompositeElementDataPrototype`.]()

As explained before, the implementation of `AutosarParameterRef` in a specific case is subject to [constr\_1173].

[constr\_1608] Existence of `rootParameterDataPrototype` [The reference `rootParameterDataPrototype` shall exist **at any time in the workflow** if and only if

- `AutosarDataType` of the `autosarParameter` is a composite data type and
- `targetDataPrototype` refers to a `DataPrototype` inside the `rootParameterDataPrototype`.

]()

Note: If the target of the `AtpInstanceRef` is an `AutosarDataPrototype` then the `rootParameterDataPrototype` shall not exist.

[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` for the case that the `targetDataPrototype` is an `ApplicationCompositeElementDataPrototype`.]()

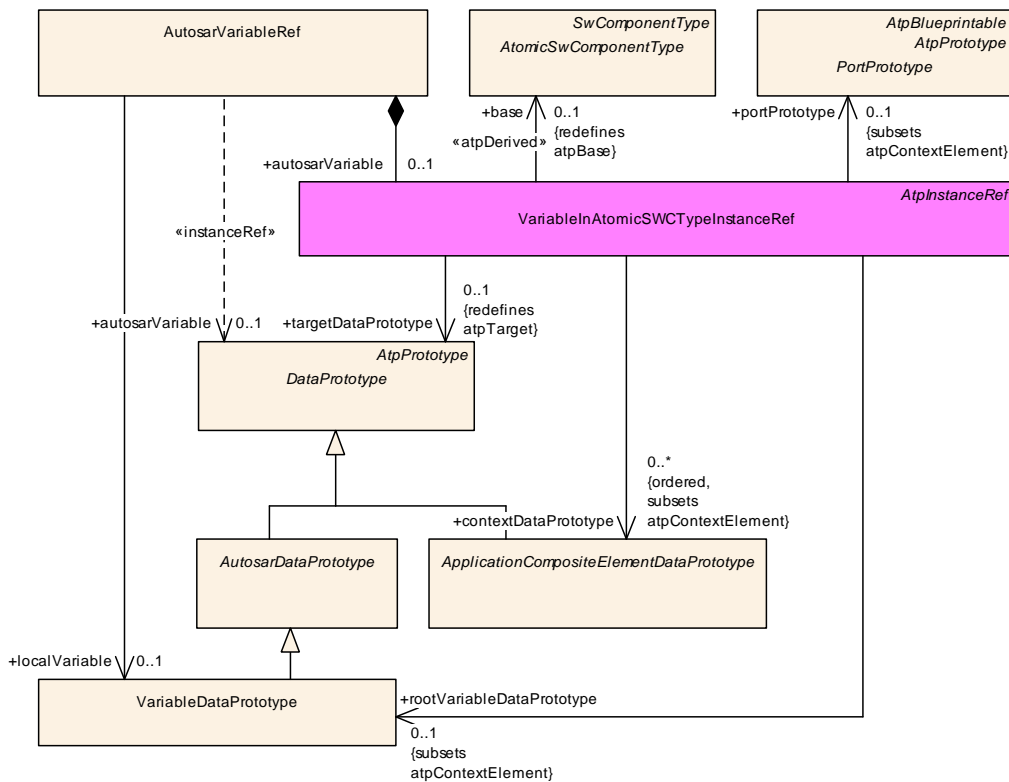


Figure 5.25: Implementation of the InstanceRef for **AutosarVariableRef**



[constr\_1609] Existence of **rootVariableDataPrototype** [The reference **rootVariableDataPrototype** shall exist **at any time in the workflow** if and only if

- the **AutosarDataType** of the **autosarVariable** is a composite data type and
- the **targetDataPrototype** refers to a **DataPrototype** inside the **rootVariableDataPrototype**.

]()

### 5.3.3.4 Access into VariableDataPrototype typed by an Implementation-DataType

The meta-class **ArVariableInImplementationDataInstanceRef**, despite the name, has formally no relationship to **AtpInstanceRef**. Therefore, the following definition applies:

[TPS\_SWCT\_01681] Context path in **ArVariableInImplementationDataInstanceRef** [The references in the roles

- **portPrototype**
- **rootVariableDataPrototype**
- ordered collection of **contextDataPrototype**
- **targetDataPrototype**

constitute the path leading from the root to the specified inner instance of a **dataElement** inside a **VariableDataPrototype** typed by an **Implementation-DataType**.]()

This relation is also depicted in Figure 5.26.

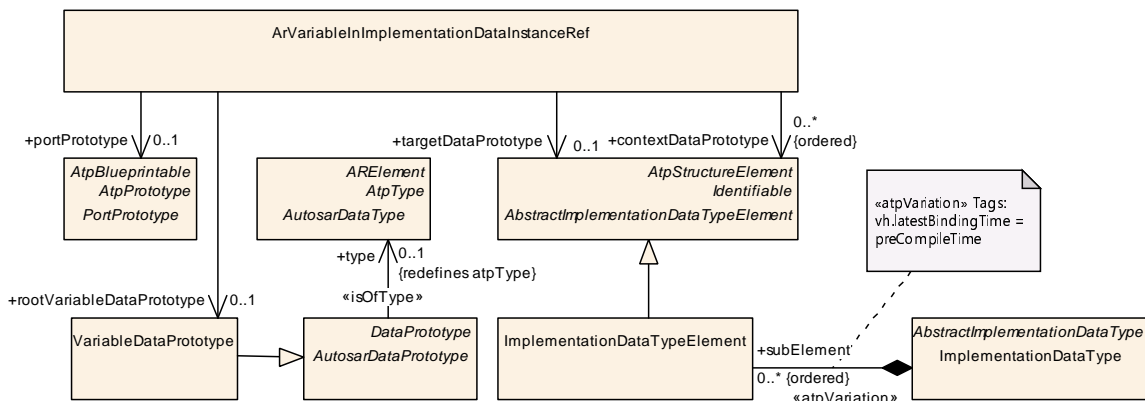
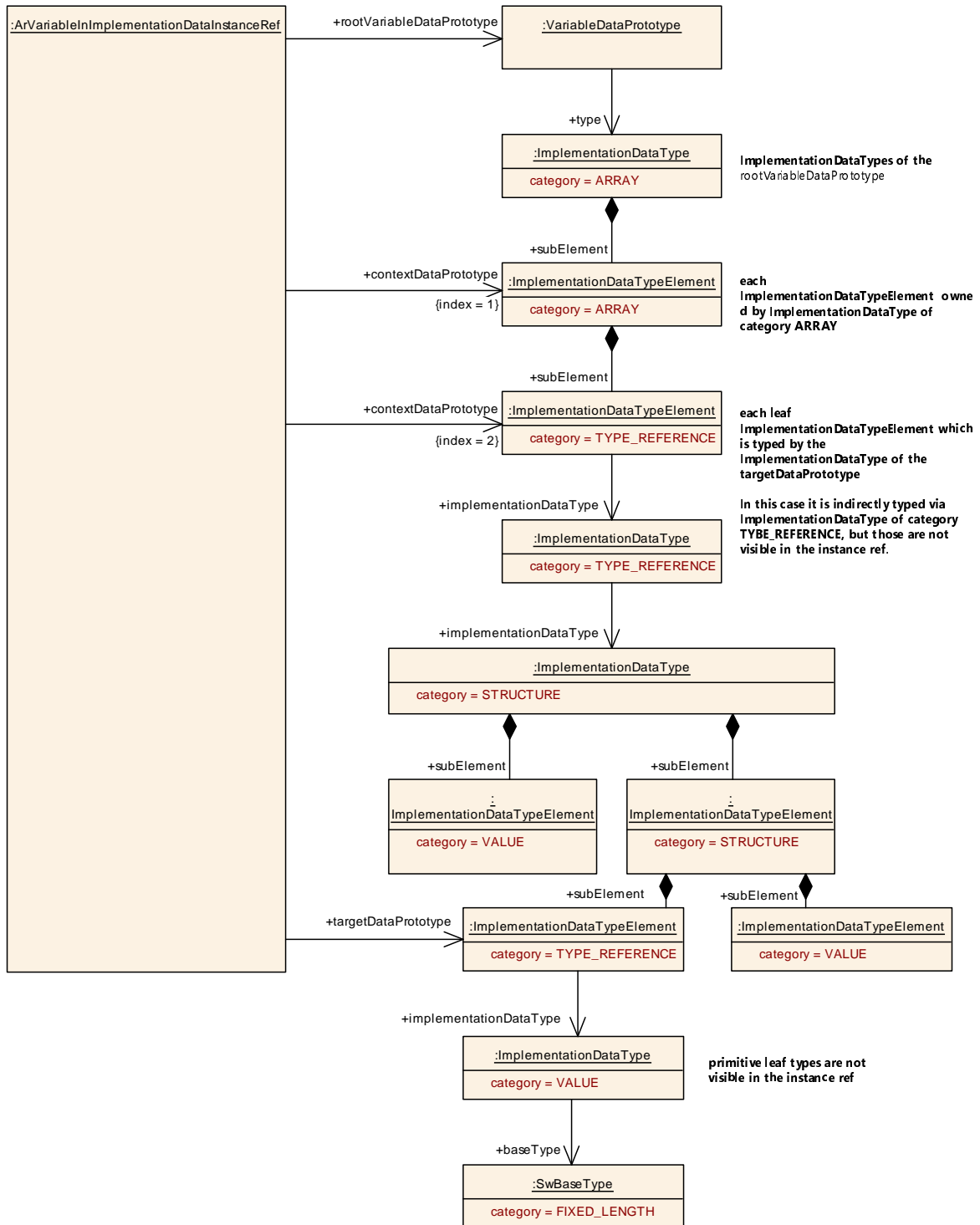


Figure 5.26: Implementation of **ArVariableInImplementationDataInstanceRef**



**Figure 5.27:** Example for the usage of `ArVariableInImplementationDataInstanceRef`

**[constr\_1423] Completeness of references `ArVariableInImplementationDataInstanceRef.contextDataPrototype`** [The reference `ArVariableInImplementationDataInstanceRef.contextDataPrototype` shall be defined for

- each *leaf* (i.e. the end of a chain of aggregating elements) `ImplementationDataTypeElement` of category `TYPE_REFERENCE` in a chain of referencing `ImplementationDataTypes` which is not the `targetDataPrototype`
- and each `ImplementationDataTypeElement` owned by an `ImplementationDataType` or `ImplementationDataTypeElement` of category `ARRAY` in a chain of referencing `ImplementationDataTypes`

starting from the `ImplementationDataTypes` of the `rootVariableDataPrototype` down to the leaf `ImplementationDataTypeElement` which is typed (directly or indirectly via `ImplementationDataType` of category `TYPE_REFERENCE`) by the `ImplementationDataType` of the `targetDataPrototype`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Figure 5.27 contains an example of a nested `ImplementationDataType` along with the application of `ArVariableInImplementationDataInstanceRef`. The example contains both cases for the definition of a `contextDataPrototype` mentioned in [constr\_1423].

**[constr\_1424] Existence of `ArVariableInImplementationDataInstanceRef`.** `contextDataPrototype` [The attribute `ArVariableInImplementationDataInstanceRef.contextDataPrototype` shall only exist **at any time in the workflow** for an `ImplementationDataTypeElement` category `TYPE_REFERENCE` or `ARRAY`.]()

Technically, it would be possible to avoid the context for a one-dimensional array in the hierarchy. The context is still required because then the rule for the existence of contexts becomes much simpler.

<b>Class</b>	<b>ArVariableInImplementationDataInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
<b>Note</b>	<p>This class represents the ability to navigate into a data element inside of an <code>VariableDataPrototype</code> which is typed by an <code>ImplementationDatatype</code>.</p> <p>Note that it shall not be used if the target is the <code>VariableDataPrototype</code> itself (e.g. if its a primitive).</p> <p>Note that this class follows the pattern of an <code>InstanceRef</code> but is not implemented based on the abstract classes because the <code>ImplementationDataType</code> isn't either, especially because <code>ImplementationDataTypeElement</code> isn't derived from <code>AtpPrototype</code>.</p>			
<b>Base</b>	<code>ARObject</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
<code>contextDataPrototype (ordered)</code>	<code>AbstractImplementationDataTypeElement</code>	*	ref	<p>This is a context in case there are subelements with explicit types. The reference has to be ordered to properly reflect the nested structure.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>
<code>portPrototype</code>	<code>PortPrototype</code>	0..1	ref	<p>This is the port providing/receiving the root of the variable</p> <p><b>Tags:</b>xml.sequenceOffset=10</p>





Class	ArVariableInImplementationDataInstanceRef			
rootVariableDataPrototype	<a href="#">VariableDataPrototype</a>	0..1	ref	This refers to the VariableDataPrototype typed by the ImplementationDatatype in which the target can be found. <b>Tags:</b> xml.sequenceOffset=20
targetDataPrototype	<a href="#">AbstractImplementationData TypeElement</a>	0..1	ref	This reference points to the target ImplementationData TypeElement. <b>Tags:</b> xml.sequenceOffset=40

**Table 5.41: ArVariableInImplementationDataInstanceRef**

**[constr\_1911] Existence of ArVariableInImplementationDataInstanceRef.targetDataPrototype** [For each ArVariableInImplementationDataInstanceRef, the reference targetDataPrototype shall exist at the time when the contract phase generation is executed.]()

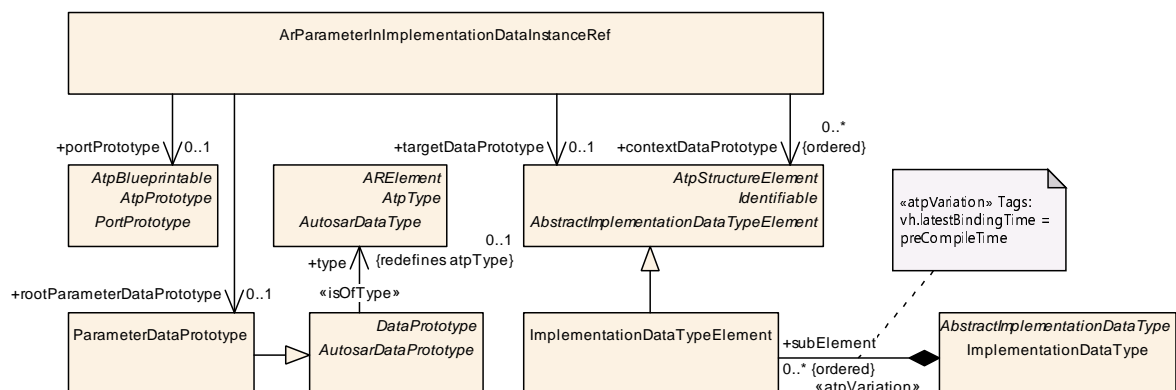
### 5.3.3.5 Access into ParameterDataPrototype typed by an Implementation-Data Type

Please note that it is also possible to access the inside of a nested ParameterDataPrototype typed by an ImplementationData Type in pretty much the same way as this is possible for a VariableDataPrototype typed by an Implementation-Data Type.

**[TPS\_SWCT\_01738] Context path in ArParameterInImplementationDataInstanceRef** [The references in the roles

- portPrototype
- rootParameterDataPrototype
- ordered collection of contextDataPrototype
- targetDataPrototype

constitute the path leading from the root to the specified inner instance of a parameter inside a ParameterDataPrototype typed by an ImplementationData Type.]()



**Figure 5.28: Implementation of ArParameterInImplementationDataInstanceRef**

This relation is also depicted in Figure 5.28.

**[constr\_1516] Completeness of references `ArParameterInImplementationDataInstanceRef.contextDataPrototype`** [The reference `ArParameterInImplementationDataInstanceRef.contextDataPrototype` shall be defined for

- each *leaf* (i.e. the end of a chain of aggregating elements) `ImplementationDataTypeElement` of category `TYPE_REFERENCE` in a chain of referencing `ImplementationDataTypes` which is not the `targetDataPrototype`
- and each `ImplementationDataTypeElement` owned by an `ImplementationDataType` or `ImplementationDataTypeElement` of category `ARRAY` in a chain of referencing `ImplementationDataTypes`

starting from the `ImplementationDataTypes` of the `rootParameterDataPrototype` down to the leaf `ImplementationDataTypeElement` which is typed (directly or indirectly via `ImplementationDataType` of category `TYPE_REFERENCE`) by the `ImplementationDataType` of the `targetDataPrototype`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1517] Existence of `ArParameterInImplementationDataInstanceRef.contextDataPrototype`** [The attribute `ArParameterInImplementationDataInstanceRef.contextDataPrototype` shall only exist **at any time in the workflow** for an `ImplementationDataTypeElement` category `TYPE_REFERENCE` or `ARRAY`.]()

Technically, it would be possible to avoid the context for a one-dimensional array in the hierarchy. The context is still required because then the rule for the existence of contexts becomes much simpler.

<b>Class</b>	<b>ArParameterInImplementationDataInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
<b>Note</b>	<p>This class represents the ability to navigate into an element inside of an <code>ParameterDataPrototype</code> typed by an <code>ImplementationDatatype</code>.</p> <p>Note that it shall not be used if the target is the <code>ParameterDataPrototype</code> itself (e.g. if the target is a primitive data type).</p> <p>Note that this class follows the pattern of an <code>InstanceRef</code> but is not implemented based on the abstract classes because the <code>ImplementationDataType</code> isn't either, especially because <code>ImplementationDataTypeElement</code> (intentionally) isn't derived from <code>AtpPrototype</code>.</p>			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
contextDataPrototype (ordered)	<a href="#">AbstractImplementationDataTypeElement</a>	*	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.
portPrototype	<a href="#">PortPrototype</a>	0..1	ref	This reference points to the <code>PortPrototype</code> providing/receiving the root of the parameter.
rootParameterDataPrototype	<a href="#">ParameterDataPrototype</a>	0..1	ref	This refers to the <code>ParameterDataPrototype</code> typed by the <code>implementationDataType</code> in which the target can be found.





Class	ArParameterInImplementationDataInstanceRef			
targetData Prototype	<a href="#">AbstractImplementation DataTypeElement</a>	0..1	ref	This reference points to the target ImplementationData TypeElement.

**Table 5.42: ArParameterInImplementationDataInstanceRef**

**[constr\_1912] Existence of reference [ArParameterInImplementationDataInstanceRef.targetDataPrototype](#)** [For each [ArParameterInImplementationDataInstanceRef](#), the reference [targetDataPrototype](#) shall exist **at the time when the contract phase generation is executed.**]()

## 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:

Attributes of SwDataDefProps	Usage For			Place of Setting										
	RTE	A2L	Other Usage	ApplicationDataType	ImplementationDataType	DataPrototype	InstantiationDataDefProps	ParameterAccess	ComSpec	SwServiceArg	FlatInstanceDescriptor	McDataInstance	SwSystemconst	PerInstanceMemory
<a href="#">additionalNativeTypeQualifier</a>	x		x	NA	D	I	NA	NA	NA	D	NA	S	NA	NA
<a href="#">annotation</a>			x	D	A	A	A	A	A	D	NA	A	D	NA
<a href="#">baseType</a>	x	x	x	NA	D	I	I	I	R	D	NA	S	M	NA
<a href="#">compuMethod</a>	x	x	x	D	AI	I	I	NA	R	I	AI	S	D	NA
<a href="#">dataConstr</a>	x	x	x	D	C	R	R	I	NA	R	NA	S	D	NA
<a href="#">displayFormat</a>		x		D	A	R	R	I	NA	R	NA	S	D	NA
<a href="#">displayPresentation</a>	x	x	x	D	A	R	R	NA	NA	NA	NA	S	NA	NA





Attributes of SwDataDefProps	Usage For			Place of Setting										
	RTE	A2L	Other Usage	ApplicationDataType	ImplementationDataType	DataPrototype	InstantiationDataDefProps	ParameterAccess	ComSpec	SwServiceArg	FlatInstanceDescriptor	McDataInstance	SwSystemconst	PerInstanceMemory
implementationDataType	x		x	NA	D	I	I	I	NA	D	NA	NA	NA	NA
invalidValue	x	x		D	A	I	I	NA	D	NA	NA	S	NA	NA
stepSize		x		D	A	A	A	A	NA	NA	A	S	NA	NA
swAddrMethod	x	x	x	D	R	R	R	NA	NA	NA	R	NA	NA	D
swAlignment	x		x	NA	D	R	R	NA	NA	NA	NA	NA	NA	NA
swBitRepresentation		x	x	NA	NA	NA	NA	NA	NA	NA	NA	D	NA	NA
swCalibrationAccess	x	x		D	R	R	R	NA	NA	R	R	S	D	NA
swCalprmAxisSet	x	x		D	NA	I	I	I	NA	NA	NA	S	NA	NA
swCalprmAxisSet.swCalprmAxis/SwAxisGrouped.swCalprmRef		x		NA	NA	NA	D	R	NA	NA	NA	S	NA	NA
swCalprmAxisSet.swCalprmAxis/SwAxisIndividual.swVariableRef		x		NA	NA	NA	D	R	NA	NA	NA	S	NA	NA
swCalprmAxisSet.swCalprmAxis/SwAxisGrouped.sharedAxisType		x		D	NA	NA	NA	NA	NA	NA	NA	S	NA	NA
swCalprmAxisSet.swCalprmAxis/SwAxisIndividual.inputVariableType		x		D	NA	NA	NA	NA	NA	NA	NA	S	NA	NA
swCalprmAxisSet/SwAxisIndividual.unit		opt.		D	NA	I	I	I	NA	I	NA	S	NA	NA
swComparisonVariable		x		NA	NA	NA	NA	D	NA	NA	NA	S	NA	NA
swDataDependency		x	x	NA	NA	D	R	NA	NA	NA	NA	S	NA	NA
swHostVariable		x	x	NA	NA	NA	NA	NA	NA	NA	NA	D	NA	NA
swImplPolicy	x		x	D	A	A	NA	NA	NA	D	NA	NA	NA	NA
swIntendedResolution			x	D <sup>15</sup>	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
swInterpolationMethod			x	D	I	R	R	R	NA	NA	NA	S	NA	NA
swIsVirtual		x		NA	NA	D	R	NA	NA	NA	NA	S	NA	NA
swPointerTargetProps			x	NA	D	I	NA	NA	NA	D	NA	NA	NA	NA
swRecordLayout	x	x	x	D	NA	I	I	I	NA	NA	NA	S	NA	NA
swRefreshTiming		x		D	R	R	R	NA	NA	R	R	R	NA	NA
swTextProps		x	x	D	I	I	I	I	NA	NA	NA	S	NA	NA
swValueBlockSize		x	x	D	I	I	I	I	NA	NA	NA	S	NA	NA
swValueBlockSizeMult		x	x	D	I	I	I	I	NA	NA	NA	S	NA	NA
unit		x	x	D	I	I	I	NA	NA	I	NA	S	D	NA
valueAxisDataType		x	x	D	I	I	I	I	NA	NA	NA	S	NA	NA

**Table 5.43: Usage of Attributes of SwDataDefProps**

Please note that this table is (by reference) a part of [constr\_1015]

<sup>15</sup> 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.

**[constr\_1015] Prioritization of `SwDataDefProps`** [The prioritization and usage of attributes of meta-class `SwDataDefProps` shall follow the restrictions given in table 5.43 at the time when the contract phase generation is executed.]()

The following settings apply in table 5.43:

**D Define** the attribute independent of 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 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 an 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.

**S** Create a “Self-contained” artifact based on the left.

Example: A `CompuMethod` defined in the context of a `System` of category `ECU_EXTRACT` is copied into the separate artifact for the `McSupportData` and references need to be updated to the copy.

Use case: Provide a `McDataGenerator` with a single, self-contained file to do its job.

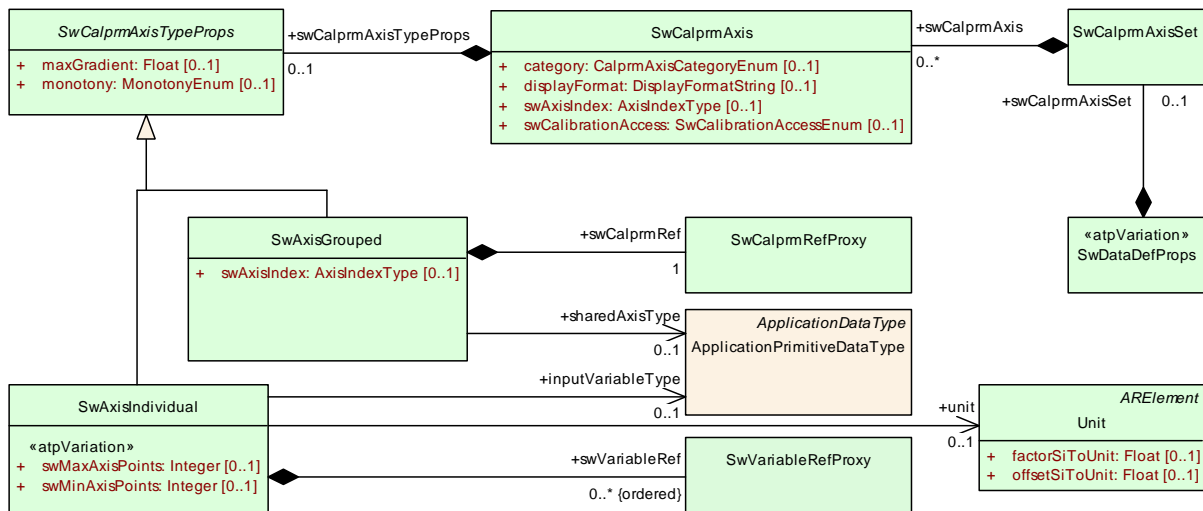


Some property names contain the term “variable” or “calprm”, this comes from historical<sup>16</sup> reasons and can be taken as some hint where the property most likely applies to.

The usage of the "/" in the table rows mentioning the content of `swCalprmAxisSet`, `swCalprmAxis`, in particular `SwAxisGrouped.swCalprmRef` and `SwAxisGrouped.sharedAxisType` resp. `SwAxisIndividual.swVariableRef`, `SwAxisIndividual.unit`, and `SwAxisIndividual.inputVariableType` represents a "shortcut" that glosses over a specific aspect of the modeling of `SwCalprmAxis` that is visible in the model but does not appear in the AUTOSAR XML schema, see Figure 5.29 (which contains all the meta-classes and roles mentioned in the table entries).

In particular, the "shortcut" affects the existence of meta-class `SwCalprmAxisTypeProps` and its aggregation in the role `SwCalprmAxis.swCalprmAxisTypeProps`. As depicted in Figure 5.29, `SwCalprmAxisTypeProps` acts as an abstract base class to both `SwAxisGrouped` and `SwAxisIndividual`.

In ARXML files that conform to the AUTOSAR XML schema, however, both `SwAxisGrouped` and `SwAxisIndividual` appear as direct child elements of `SwCalprmAxis`.



**Figure 5.29: Modeling of `SwAxisGrouped` and `SwAxisIndividual`**

This difference between meta-model and AUTOSAR XML schema is explained by the existence of a set of tags at the aggregation `SwCalprmAxis.swCalprmAxisTypeProps`. The details of how these tags impact the schema generation are explained in the *TPS XML Schema Production Rules* [25].

To summarize, the "shortcut" in the table rows simply approximates the situation in ARXML instead of reflecting the actual modeling in the AUTOSAR meta-model.

<sup>16</sup>In the beginning of ASAM and MSR, measurements and calibration parameters (characteristics) were separated and the properties were merged over time.

<b>Class</b>	<<atpVariation>> <b>SwDataDefProps</b>			
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties			
<b>Note</b>	<p>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.</p> <p>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.</p> <p>SwDataDefProps covers various aspects:</p> <ul style="list-style-type: none"> <li>• 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 Data Types in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet</li> <li>• Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddrMethod, swPointerTargetProps, baseType, implementationDataType and additionalNativeTypeQualifier</li> <li>• Access policy for the MCD system, mainly expressed by swCalibrationAccess</li> <li>• Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue</li> <li>• Code generation policy provided by swRecordLayout</li> </ul> <p><b>Tags:</b>vh.latestBindingTime=codeGenerationTime</p>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
additionalNativeTypeQualifier	<a href="#">NativeDeclarationString</a>	0..1	attr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=235</p>
annotation	<a href="#">Annotation</a>	*	aggr	<p>This aggregation allows to add annotations (yellow pads ...) related to the current data object.</p> <p><b>Tags:</b>                      xml.roleElement=true                      xml.roleWrapperElement=true                      xml.sequenceOffset=20                      xml.typeElement=false                      xml.typeWrapperElement=false</p>
baseType	<a href="#">SwBaseType</a>	0..1	ref	<p>Base type associated with the containing data object.</p> <p><b>Tags:</b>xml.sequenceOffset=50</p>
compuMethod	<a href="#">CompuMethod</a>	0..1	ref	<p>Computation method associated with the semantics of this data object.</p> <p><b>Tags:</b>xml.sequenceOffset=180</p>
dataConstr	<a href="#">DataConstr</a>	0..1	ref	<p>Data constraint for this data object.</p> <p><b>Tags:</b>xml.sequenceOffset=190</p>
displayFormat	<a href="#">DisplayFormatString</a>	0..1	attr	<p>This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system.</p> <p><b>Tags:</b>xml.sequenceOffset=210</p>
displayPresentation	<a href="#">DisplayPresentationEnum</a>	0..1	attr	<p>This attribute controls the presentation of the related data for measurement and calibration tools.</p>





Class	<<atpVariation>> SwDataDefProps			
implementation DataType	<a href="#">AbstractImplementation DataType</a>	0..1	ref	<p>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</p> <ul style="list-style-type: none"> <li>• redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype</li> <li>• the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly</li> <li>• the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly</li> <li>• the data type of an SwServiceArg, if it does not refer to a base type directly</li> </ul> <p><b>Tags:</b>xml.sequenceOffset=215</p>
invalidValue	<a href="#">ValueSpecification</a>	0..1	aggr	<p>Optional value to express invalidity of the actual data element.</p> <p><b>Tags:</b>xml.sequenceOffset=255</p>
stepSize	Float	0..1	attr	<p>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.</p>
swAddrMethod	<a href="#">SwAddrMethod</a>	0..1	ref	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>
swAlignment	<a href="#">AlignmentType</a>	0..1	attr	<p>The attribute describes the intended typical alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced Sw AddrMethod.</p> <p><b>Tags:</b>xml.sequenceOffset=33</p>
swBit Representation	<a href="#">SwBitRepresentation</a>	0..1	aggr	<p>Description of the binary representation in case of a bit variable.</p> <p><b>Tags:</b>xml.sequenceOffset=60</p>
swCalibration Access	<a href="#">SwCalibrationAccess Enum</a>	0..1	attr	<p>Specifies the read or write access by MCD tools for this data object.</p> <p><b>Tags:</b>xml.sequenceOffset=70</p>
swCalprmAxis Set	<a href="#">SwCalprmAxisSet</a>	0..1	aggr	<p>This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters.</p> <p><b>Tags:</b>xml.sequenceOffset=90</p>
swComparison Variable	<a href="#">SwVariableRefProxy</a>	*	aggr	<p>Variables used for comparison in an MCD process.</p> <p><b>Tags:</b> xml.sequenceOffset=170 xml.typeElement=false</p>
swData Dependency	<a href="#">SwDataDependency</a>	0..1	aggr	<p>Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system).</p> <p><b>Tags:</b>xml.sequenceOffset=200</p>





Class	<<atpVariation>> SwDataDefProps			
swHostVariable	<a href="#">SwVariableRefProxy</a>	0..1	aggr	<p>Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects.</p> <p><b>Tags:</b> xml.sequenceOffset=220 xml.typeElement=false</p>
swImplPolicy	<a href="#">SwImplPolicyEnum</a>	0..1	attr	<p>Implementation policy for this data object.</p> <p><b>Tags:</b>xml.sequenceOffset=230</p>
swIntendedResolution	Numerical	0..1	attr	<p>The purpose of this element is to describe the requested quantization of data objects early on in the design process.</p> <p>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).</p> <p>In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution.</p> <p>The resolution is specified in the physical domain according to the property "unit".</p> <p><b>Tags:</b>xml.sequenceOffset=240</p>
swInterpolationMethod	Identifier	0..1	attr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=250</p>
swIsVirtual	Boolean	0..1	attr	<p>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 .</p> <p><b>Tags:</b>xml.sequenceOffset=260</p>
swPointerTargetProps	<a href="#">SwPointerTargetProps</a>	0..1	aggr	<p>Specifies that the containing data object is a pointer to another data object.</p> <p><b>Tags:</b>xml.sequenceOffset=280</p>
swRecordLayout	<a href="#">SwRecordLayout</a>	0..1	ref	<p>Record layout for this data object.</p> <p><b>Tags:</b>xml.sequenceOffset=290</p>
swRefreshTiming	<a href="#">MultidimensionalTime</a>	0..1	aggr	<p>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.</p> <p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=300</p>
swTextProps	<a href="#">SwTextProps</a>	0..1	aggr	<p>the specific properties if the data object is a text object.</p> <p><b>Tags:</b>xml.sequenceOffset=120</p>
swValueBlockSize	Numerical	0..1	attr	<p>This represents the size of a Value Block</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=80</p>





Class	<<atpVariation>> SwDataDefProps			
swValueBlockSizeMult (ordered)	Numerical	*	attr	<p>This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension.</p> <p>The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on.</p> <p>For one-dimensional value blocks the attribute swValueBlockSize shall be used and this attribute shall not exist.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
unit	Unit	0..1	ref	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=350</p>
valueAxisDataType	ApplicationPrimitiveDataType	0..1	ref	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=355</p>

**Table 5.44: SwDataDefProps**

<b>Primitive</b>	<b>NativeDeclarationString</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
<b>Note</b>	<p>This string contains a native data declaration of a data type in a programming language. It is basically a string, but white-space shall be preserved.</p> <p><b>Tags:</b>                      xml.xsd.customType=NATIVE-DECLARATION-STRING                      xml.xsd.type=string                      xml.xsd.whiteSpace=preserve</p>

**Table 5.45: NativeDeclarationString**

Class	SwBitRepresentation			
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
Attribute	Type	Mult.	Kind	Note
bitPosition	Integer	0..1	attr	<p>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).</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>
numberOfBits	Integer	0..1	attr	<p>Number of bits allocated by a "bit data object" within its host data object.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>

**Table 5.46: SwBitRepresentation**

<b>Primitive</b>	<b>DisplayFormatString</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
<b>Note</b>	<p>This is a display format specifier for the display of values e.g. in documents or in measurement and calibration systems.</p> <p>The display format specifier is a subset of the ANSI C printf specifiers with the following form: % [flags] [width] [.prec] type character</p> <p>For more details refer to "ASAM-HarmonizedDataObjects-V1.1.pdf" chapter 13.3.2 DISPLAY OF DATA.</p> <p>Due to the numerical nature of value settings, only the following type characters are allowed:</p> <ul style="list-style-type: none"> <li>• d: Signed decimal integer</li> <li>• i: Signed decimal integer</li> <li>• o: Unsigned octal integer</li> <li>• u: Unsigned decimal integer</li> <li>• x: Unsigned hexadecimal integer, using "abcdef"</li> <li>• X: Unsigned hexadecimal integer, using "ABCDEF"</li> <li>• 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 -</li> <li>• E: Identical to the e format except that E rather than e introduces the exponent</li> <li>• 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</li> <li>• 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</li> <li>• G: Identical to the g format, except that E, rather than e, introduces the exponent (where appropriate)</li> </ul> <p><b>Tags:</b> xml.xsd.customType=DISPLAY-FORMAT-STRING xml.xsd.pattern=%[ \-+#]?[0-9]*(\.[0-9]+)?[diouxXfeEgGcs] xml.xsd.type=string</p>

**Table 5.47: DisplayFormatString**

<b>Class</b>	<b>Annotation</b>			
<b>Package</b>	M2::MSR::Documentation::Annotation			
<b>Note</b>	This is a plain annotation which does not have further formal data.			
<b>Base</b>	ARObject, <a href="#">GeneralAnnotation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 5.48: Annotation**

**[constr\_1244] DataPrototypes used in application software shall not be typed by C enums** [A [ImplementationDataType](#) that is used to type a [DataPrototype](#) owned by an [AtomicSwComponentType](#) shall not set [swDataDefProps.additionalNativeTypeQualifier](#) to `enum` **at the time when the contract phase generation is executed.**]()

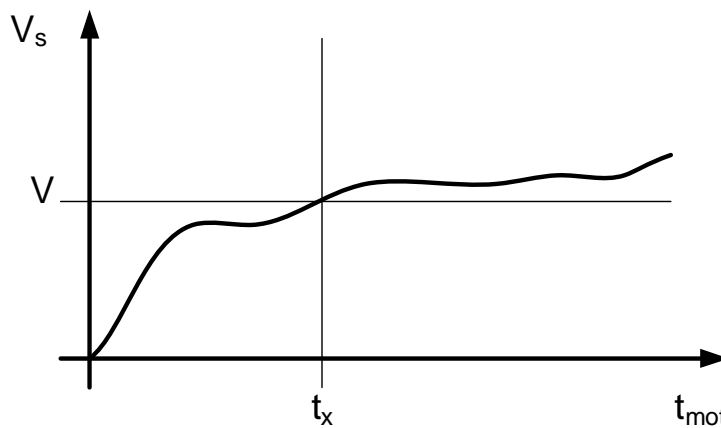
**[TPS\_SWCT\_01272] Semantics of swComparisonVariable** [Please note that [swComparisonVariables](#) shall be displayed in the MCD system on the ordinate in a curve.

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 providing 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.]()



**Figure 5.30: Explanation of `swComparisonVariable`**

The meaning behind `swComparisonVariable` is depicted in Figure 5.30. 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.

<i>Enumeration</i>	<b>SwCalibrationAccessEnum</b>
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties
<b>Note</b>	Determines the access rights to a data object w.r.t. measurement and calibration.
<b>Literal</b>	<b>Description</b>
notAccessible	The element will not be accessible via MCD tools, i.e. will not appear in the ASAP file. <b>Tags:</b> atp.EnumerationLiteralIndex=0
readOnly	The element will only appear as read-only in an ASAP file. <b>Tags:</b> atp.EnumerationLiteralIndex=1
readWrite	The element will appear in the ASAP file with both read and write access. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 5.49: 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 5.43.

**[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 [6]. 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 [RunnableEntity](#)s and show the link to an actual ECU implementation.

<i>Enumeration</i>	<b>SwImplPolicyEnum</b>
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties
<b>Note</b>	Specifies the implementation strategy with respect to consistency mechanisms of variables.
<b>Literal</b>	<b>Description</b>
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. <b>Tags:</b> atp.EnumerationLiteralIndex=0
fixed	This data element is fixed. In particular this indicates, that it might also be implemented e.g. as in place data, (#DEFINE). <b>Tags:</b> atp.EnumerationLiteralIndex=1
measurementPoint	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. <b>Tags:</b> atp.EnumerationLiteralIndex=2
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. <b>Tags:</b> atp.EnumerationLiteralIndex=3
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. <b>Tags:</b> atp.EnumerationLiteralIndex=4

**Table 5.50: SwImplPolicyEnum**

**[TPS\_SWCT\_01275] values of the attribute [swImplPolicy](#) are restricted depending on the context** [The values of the attribute [swImplPolicy](#) are restricted (summarized in table [5.51](#)) depending on the context. This restriction reflects the fact that not all possible implementation strategies are useful or supported for all kinds of [DataPrototypes](#).]()

The restrictions summarized in table [5.51](#) are formalized in a set of constraints below the table.

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 SwImplPolicyEnum	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
<b>const</b>	NA	NA	NA	NA	NA	NA	NA	x	NA	x	x	x	NA	x
<b>fixed</b>	NA	NA	NA	NA	NA	NA	NA	x	NA	NA	NA	x	NA	NA
<b>measurementPoint</b>	x	NA	NA	NA	NA	x	x	NA	NA	NA	NA	NA	NA	NA
<b>queued</b>	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
<b>standard</b>	x	x	x	x	x	x	x	x	x	x	x	x	x	x

**Table 5.51: Allowed attributes values for swImplPolicy vs. DataPrototypes and their roles**

The following settings apply in table 5.51:

**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 value of attribute swImplPolicy of a VariableDataPrototype owned by a SenderReceiverInterface shall be either standard, queued, or measurementPoint.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2036] swImplPolicy for VariableDataPrototype in NvDataInterface** [The overriding value of attribute swImplPolicy of a VariableDataPrototype owned by a NvDataInterface shall be standard.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2037] swImplPolicy for VariableDataPrototype in the role ramBlock** [The overriding value of attribute swImplPolicy of a VariableDataPrototype aggregated in the role NvBlockDescriptor.ramBlock shall be standard.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2038] swImplPolicy for VariableDataPrototype in the role implicitInterRunnableVariable** [The overriding value of attribute `swImplPolicy` of a `VariableDataPrototype` aggregated in the role `SwcInternalBehavior.implicitInterRunnableVariable` shall be `standard`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2039] swImplPolicy for VariableDataPrototype in the role explicitInterRunnableVariable** [The overriding value of attribute `swImplPolicy` of a `VariableDataPrototype` aggregated in the role `SwcInternalBehavior.explicitInterRunnableVariable` shall be `standard`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2040] swImplPolicy for VariableDataPrototype in the role arTypedPerInstanceMemory** [The overriding value of attribute `swImplPolicy` of a `VariableDataPrototype` aggregated in the role `SwcInternalBehavior.arTypedPerInstanceMemory` shall be `standard` or `measurementPoint`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2041] swImplPolicy for VariableDataPrototype in the role staticMemory** [The overriding value of attribute `swImplPolicy` of a `VariableDataPrototype` aggregated in the role `InternalBehavior.staticMemory` shall be `standard` or `measurementPoint`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2042] swImplPolicy for ParameterDataPrototype in ParameterInterface** [The overriding value of attribute `swImplPolicy` of a `ParameterDataPrototype` owned by a `ParameterInterface` shall be either `standard`, `const`, or `fixed`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2043] swImplPolicy for ParameterDataPrototype in the role romBlock** [The overriding value of attribute `swImplPolicy` of a `ParameterDataPrototype` aggregated in the role `NvBlockDescriptor.romBlock` shall be `standard`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2044] swImplPolicy for ParameterDataPrototype in the role sharedParameter** [The overriding value of attribute `swImplPolicy` of a `ParameterDataPrototype` aggregated in the role `SwcInternalBehavior.sharedParameter` shall be `standard` or `const`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2045] swImplPolicy for ParameterDataPrototype in the role perInstanceParameter** [The overriding value of attribute `swImplPolicy` of a `ParameterDataPrototype` in the role `SwcInternalBehavior.perInstanceParameter` shall be `standard` or `const`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2046] swImplPolicy for ParameterDataPrototype in the role constantMemory** [The overriding value of attribute `swImplPolicy` of a `ParameterDataPrototype` aggregated in the role `InternalBehavior.constantMemory` shall be `standard`, `const`, or `fixed`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2047] swImplPolicy for ArgumentDataPrototype** [The overriding value of attribute `swImplPolicy` of an `ArgumentDataPrototype` shall be `standard`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2048] swImplPolicy for SwServiceArg** [The overriding value of attribute `swImplPolicy` of a `SwServiceArg` shall be `standard` or `const`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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`, the default value `standard` applies.]()

Please note that the locations listed in “Place of Setting” for `SwDataDefProps` are described in Table 5.43.

## 5.4.2 Invalid Value

The diagram 5.5 shows that in addition to the semantics defined through the `computeMethod` (explained below in chapter 5.5.1), also an `invalidValue` can be specified. This is a requirement of the VFB [3], allowing expressing which specific value is used to indicate invalidation.

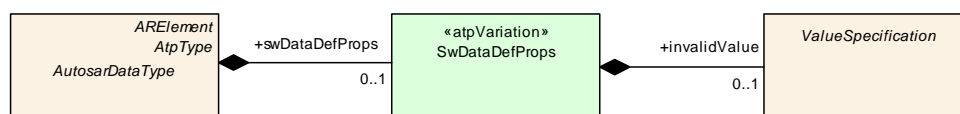


Figure 5.31: Invalid value

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 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 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` **at the time when the contract phase generation is executed.**.]()

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` **at the time when the contract phase generation is executed.**.]()

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 be either a compatible `NumericalValueSpecification`, `TextValueSpecification` (caution, [constr\_1284] applies) or a `ConstantReference` that in turn points to a compatible `ValueSpecification`.

This rule shall be imposed **at the time when the contract phase generation is executed.**.]()

**[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` **at any time in the workflow.**.]()

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` **at the time when the contract phase generation is executed.**]()

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Please note that several attributes of meta-class `CompuScale` can be taken to describe the actual value. It is therefore necessary to clarify what happens if several of these attributes exist within the context of one `CompuScale`. This clarification can be found in **[TPS\_SWCT\_01696]**.

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Please note that [constr\_1302], in general, leaves room for the definition of an invalid value for a `DataPrototype` typed by a `Wrapped Union Data Type` because it demands the existence of a primitive element that has an `invalidValue`. In the case of a `Wrapped Union Data Type`, the primitive element could be the `Member Selector`, and thus [constr\_1302] would technically be fulfilled.

On the one hand, it does not make sense to just define an invalid value for the `Member Selector` from the semantic point of view. On the other hand, the actual payload may not even have an invalid value according to [constr\_1009] or [constr\_1288], respectively.

In order to simplify the situation and make a clear statement, [constr\_1446] has been defined.

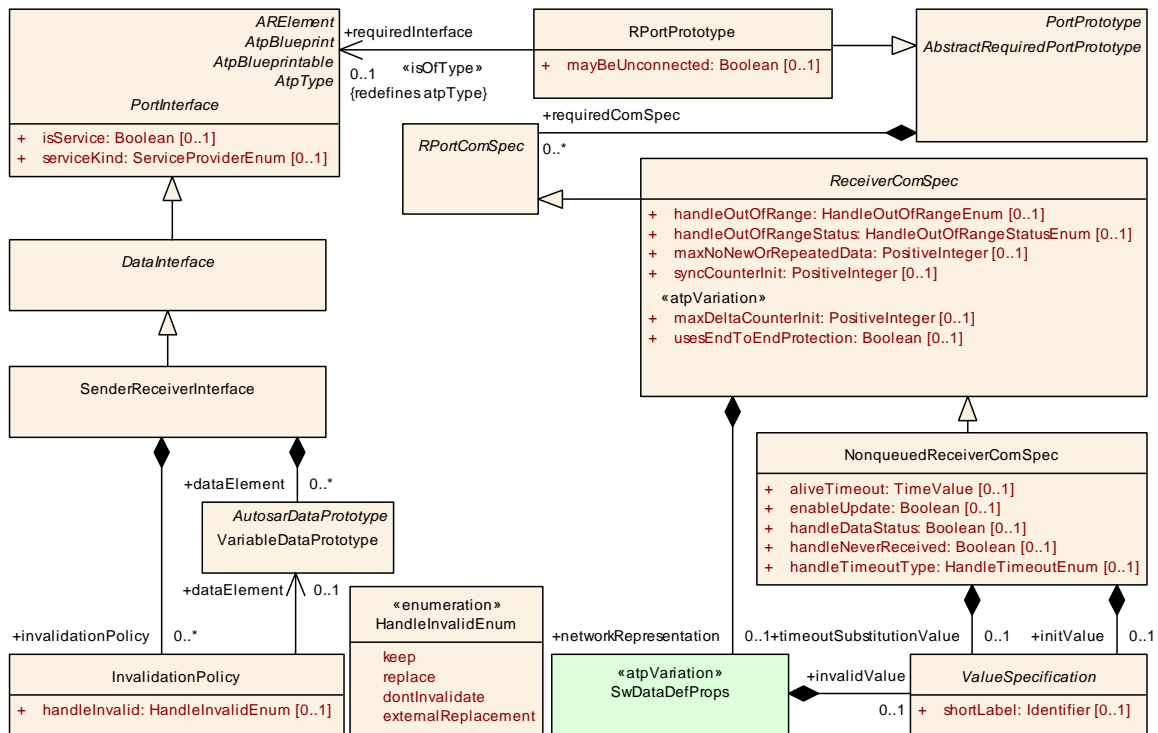
**[constr\_1446] No definition of `invalidValue` for a `Wrapped Union Data Type`**  
[The definition of an `invalidValue` for a `DataPrototype` typed by a `Wrapped Union Data Type` is not supported **at any time in the workflow.**]()

**[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 **at the time when the contract phase generation is executed.**]()

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`.





**Figure 5.32: Relationships required to consider the `invalidValue`**

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 5.32.

**[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` at the time when the contract phase generation is executed.]()

**[constr\_1282] Restriction concerning the usage of `RuleBasedValueSpecification` or a `ReferenceValueSpecification` for the specification of an `invalidValue`** [The aggregation of a `RuleBasedValueSpecification` or a `ReferenceValueSpecification` for the definition of a `ApplicationPrimitiveDataType.swDataDefProps.invalidValue` is not supported at the time when the contract phase generation is executed.]()

### 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 an 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`:

<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 5.52: Supported combinations of `swImplPolicy` and `swCalibrationAccess`**

**[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 5.52 defines the supported combinations at the time when the contract phase generation is executed of `swImplPolicy` and `swCalibrationAccess` attribute setting.]()

**[constr\_1018] `measurementPoint` shall not be referenced by a `VariableAccess` aggregated by `RunnableEntity` in the role `dataReadAccess`** [Due to the nature of `dataElements` characterized by setting the `swImplPolicy` to `measurementPoint`, such `dataElements` shall not be referenced by a `VariableAccess` aggregated by `RunnableEntity` in the role `dataReadAccess` at the time when the contract phase generation is executed.]()

#### 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`.

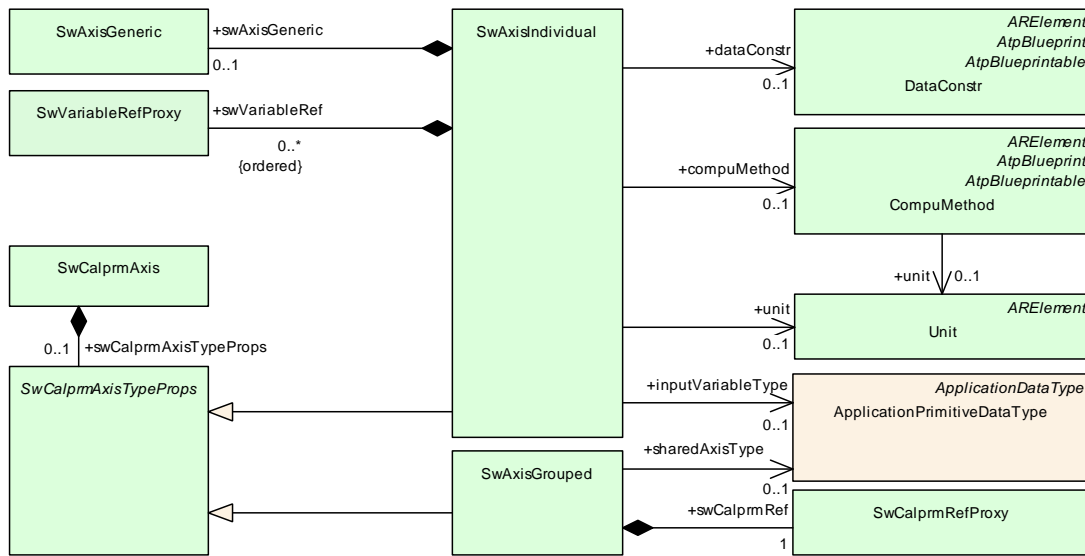
The axis description itself is defined by the meta-model element `SwCalprmAxisSet` aggregating the appropriate number of `SwCalprmAxisTypeProps`.

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`).

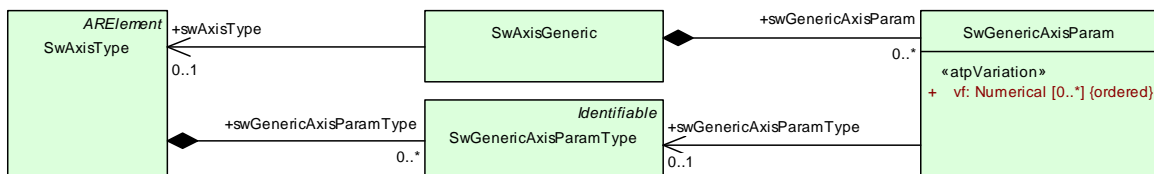
The latter is used to share axis points by several characteristic tables. Figure 5.33 shows an overview on the relevant meta-model elements.

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.

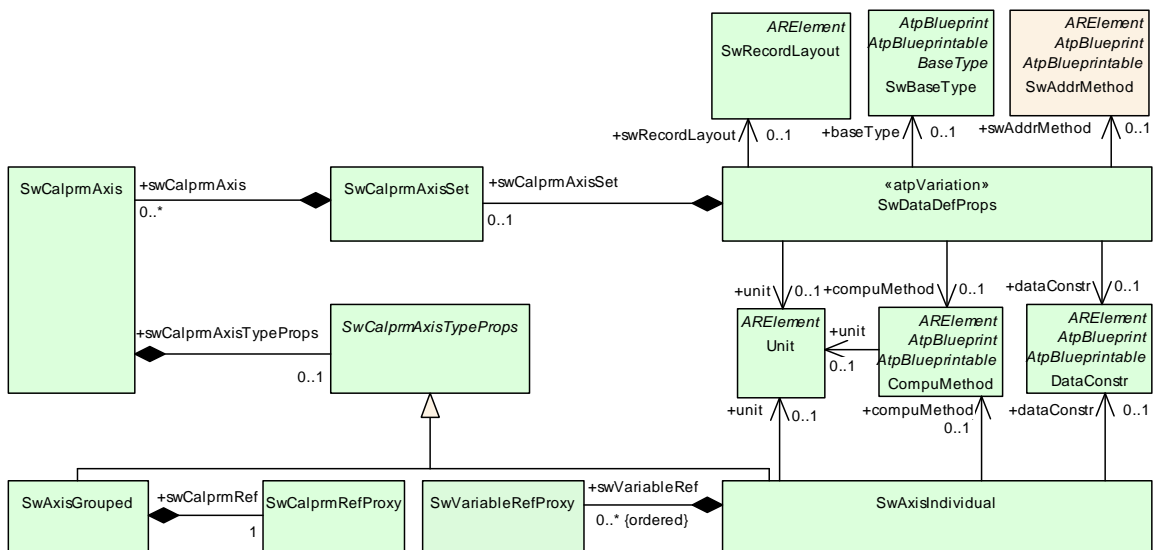


**Figure 5.33: Overview on the Meta-Model for Axis Description**



**Figure 5.34: Overview on a Generic Axis**

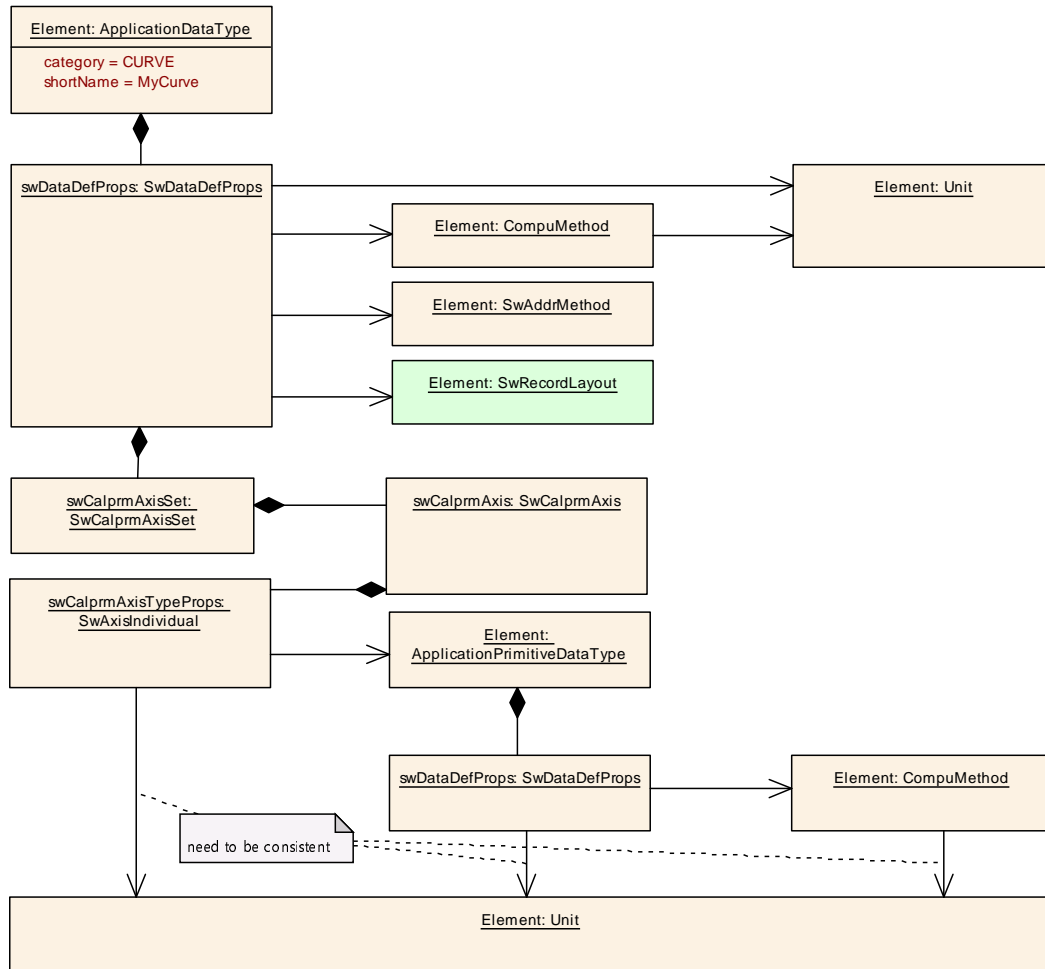
Figure 5.35 shows how an individual axis is represented by the meta-model. The corresponding M1 Model is illustrated in Figure 5.36. The `SwAxisIndividual` references value-models to account the minimum and the maximum number of axis values as well as the number of axis points.



**Figure 5.35: Meta-Model Elements used for a Curve**

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 section 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 5.36: Illustration of a Curve in M1**

<b>Class</b>	<b>SwCalprmAxisSet</b>			
<b>Package</b>	M2::MSR::DataDictionary::CalibrationParameter			
<b>Note</b>	This element specifies the input parameter axes (abscissas) of parameters (and variables, if these are used adaptively).			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





<b>Class</b>	<b>SwCalprmAxisSet</b>			
swCalprmAxis	<a href="#">SwCalprmAxis</a>	*	aggr	One axis belonging to this SwCalprmAxisSet <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

**Table 5.53: SwCalprmAxisSet**

<b>Class</b>	<b>SwCalprmAxis</b>			
<b>Package</b>	M2::MSR::DataDictionary::CalibrationParameter			
<b>Note</b>	This element specifies an individual input parameter axis (abscissa).			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
category	<a href="#">CalprmAxisCategoryEnum</a>	0..1	attr	This property specifies the category of a particular axis. <b>Tags:</b> xml.sequenceOffset=30
displayFormat	<a href="#">DisplayFormatString</a>	0..1	attr	This property specifies how the axis values shall be displayed e.g. in documents or in measurement and calibration tools. <b>Tags:</b> xml.sequenceOffset=100
swAxisIndex	<a href="#">AxisIndexType</a>	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". <b>Tags:</b> xml.sequenceOffset=20
swCalibrationAccess	<a href="#">SwCalibrationAccessEnum</a>	0..1	attr	Describes the applicability of parameters and variables. <b>Tags:</b> xml.sequenceOffset=90
swCalprmAxisTypeProps	<a href="#">SwCalprmAxisTypeProps</a>	0..1	aggr	specific properties depending on the type of the axis. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=40 xml.typeElement=true xml.typeWrapperElement=false

**Table 5.54: SwCalprmAxis**

<b>Enumeration</b>	<b>CalprmAxisCategoryEnum</b>
<b>Package</b>	M2::MSR::DataDictionary::CalibrationParameter
<b>Note</b>	This enum specifies the possible values of the category property within SwCalprmAxis.
<b>Literal</b>	<b>Description</b>
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. <b>Tags:</b> atp.EnumerationLiteralIndex=0 xml.name=COM_AXIS





Enumeration	CalprmAxisCategoryEnum
fixAxis	<p>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.</p> <p><b>Tags:</b>                      atp.EnumerationLiteralIndex=4                      xml.name=FIX_AXIS</p>
resAxis	<p>RES_AXIS is also an shared axis like COM_AXIS, the difference is that this kind of axis can be used for rescaling.</p> <p><b>Tags:</b>                      atp.EnumerationLiteralIndex=6                      xml.name=RES_AXIS</p>
stdAxis	<p>STD_AXIS means that input and output axis definition are stored within this CURVE, MAP, CUBOID, CUBE_4, and CUBE_5.</p> <p>There is no shared or calculated axis.</p> <p><b>Tags:</b>                      atp.EnumerationLiteralIndex=8                      xml.name=STD_AXIS</p>

**Table 5.55: CalprmAxisCategoryEnum**

<b>Class</b>	<b>SwCalprmAxisTypeProps</b> (abstract)			
<b>Package</b>	M2::MSR::DataDictionary::CalibrationParameter			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	SwAxisGrouped, SwAxisIndividual			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
maxGradient	Float	0..1	attr	This attribute defines the maximum permissible gradient for an adjustable object (curve, map or cuboid) with respect to a specific axis. MaxGrad = maximum( absolute((Value i,k - Value i-1,k)/(Axis Point i - Axis Point i-1)) )
monotony	MonotonyEnum	0..1	attr	This attribute specifies the monotony constraint for an adjustable object (curve, map or cuboid) with respect to a specific axis. This information can be used by MCD system to verify whether the monotony constraint is fulfilled and to prevent from changes violating the constraint.

**Table 5.56: SwCalprmAxisTypeProps**

<b>Class</b>	<b>SwAxisIndividual</b>			
<b>Package</b>	M2::MSR::DataDictionary::Axis			
<b>Note</b>	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).			
<b>Base</b>	ARObject, SwCalprmAxisTypeProps			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Tags:</b> xml.sequenceOffset=30





Class	SwAxisIndividual			
dataConstr	<a href="#">DataConstr</a>	0..1	ref	Refers to constraints, e.g. for plausibility checks. <b>Tags:</b> xml.sequenceOffset=80
inputVariableType	<a href="#">ApplicationPrimitiveDataType</a>	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	<a href="#">SwAxisGeneric</a>	0..1	aggr	this specifies the properties of a generic axis if applicable. <b>Tags:</b> xml.sequenceOffset=90
swMaxAxisPoints	Integer	0..1	attr	Maximum number of base points contained in the axis of a map or curve. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
swMinAxisPoints	Integer	0..1	attr	Minimum number of base points contained in the axis of a map or curve. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=70
swVariableRef (ordered)	<a href="#">SwVariableRefProxy</a>	*	aggr	Refers to input variables of the axis. It is possible to specify more than one variable. Here the following is valid: <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul> <p>In AUTOSAR this ensured by the constraint, that the referenced input variables shall use a type compatible to "inputVariableType".</p> <ul style="list-style-type: none"> <li>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.</li> </ul> <p>These variables can be displayed simultaneously by MCD systems (adjustment systems), enabling operating points to be shown in the curves.</p> <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
unit	<a href="#">Unit</a>	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. <b>Tags:</b> xml.sequenceOffset=40

**Table 5.57: SwAxisIndividual**



<b>Class</b>	<b>SwAxisGeneric</b>			
<b>Package</b>	M2::MSR::DataDictionary::Axis			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swAxisType	SwAxisType	0..1	ref	Associated axis calculation strategy. <b>Tags:</b> xml.sequenceOffset=20
swGenericAxisParam	SwGenericAxisParam	*	aggr	Specific parameter of a generic axis. <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false

**Table 5.58: SwAxisGeneric**

<b>Class</b>	<b>SwAxisType</b>			
<b>Package</b>	M2::MSR::DataDictionary::Axis			
<b>Note</b>	This meta-class represents a specific axis calculation strategy. No formal specification is given, due to the fact that it is possible to use arbitrary algorithms for calculating axis-points. Instead, the algorithm is described verbally but the parameters are specified formally with respect to their names and constraints. As a result, SwAxisType mainly reserves appropriate keywords. <b>Tags:</b> atp.recommendedPackage=SwAxisTypes			
<b>Base</b>	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swGenericAxisDesc	DocumentationBlock	0..1	aggr	Associated axis description in textual form. <b>Tags:</b> xml.sequenceOffset=20
swGenericAxisParamType	SwGenericAxisParamType	*	aggr	Parameters for this calculation algorithm. <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

**Table 5.59: SwAxisType**

<b>Class</b>	<b>SwGenericAxisParam</b>			
<b>Package</b>	M2::MSR::DataDictionary::Axis			
<b>Note</b>	This meta-class describes a specific parameter of a generic axis. The name of the parameter is defined through a reference to a parameter type defined on a corresponding axis type. The value of the parameter is given here in case that it is not changeable during calibration. Example is shift / offset in a fixed axis.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	SwGenericAxisParam			
swGenericAxisParamType	<a href="#">SwGenericAxisParamType</a>	0..1	ref	Parameter type defined on a corresponding axis type. References can only be made to axis parameters types which are defined within the referenced axis type. <b>Tags:</b> xml.sequenceOffset=20
vf (ordered)	Numerical	*	attr	This attribute represents the value of the generic axis parameter. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false

**Table 5.60: SwGenericAxisParam**

Class	SwGenericAxisParamType			
Package	M2::MSR::DataDictionary::Axis			
Note	<p>This meta-class describes a generic axis parameter type, namely:</p> <ul style="list-style-type: none"> <li>Plausibility checks can be specified via dataConstr.</li> <li>Textual description (desc), as a formal description is not of any use, due to the large variety of possibilities.</li> <li>If this parameter contains structures, these can be simulated through the recursive use of SwGenericAxisParamTypes.</li> </ul>			
Base	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
dataConstr	<a href="#">DataConstr</a>	0..1	ref	This reference denoted data constraints applicable to the generic axis parameter. <b>Tags:</b> xml.sequenceOffset=20

**Table 5.61: SwGenericAxisParamType**

Class	SwAxisGrouped			
Package	M2::MSR::DataDictionary::Axis			
Note	An SwAxisGrouped is an axis which is shared between multiple calibration parameters.			
Base	ARObject, <a href="#">SwCalprmAxisTypeProps</a>			
Attribute	Type	Mult.	Kind	Note
sharedAxisType	<a href="#">ApplicationPrimitiveDataType</a>	0..1	ref	This is the datatype of the calibration parameter providing the shared axis.





Class	SwAxisGrouped			
swAxisIndex	<a href="#">AxisIndexType</a>	0..1	attr	<p>Describes which axis of the referenced calibration parameter provides the values for the group axis. The index satisfies the following convention:</p> <ul style="list-style-type: none"> <li>• 0 = value axis. in this case, the interpolation result of the referenced parameter is used as a base point index.</li> <li>• 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.</li> </ul> <p><b>Tags:</b>xml.sequenceOffset=20</p>
swCalprmRef	<a href="#">SwCalprmRefProxy</a>	1	aggr	<p>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.</p> <p><b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false</p>

**Table 5.62: SwAxisGrouped**

#### 5.4.4.1 Specification of grouped Axes

Please note that [SwAxisGrouped](#) has a dual nature in that it is used to

- define the data type of the shared axis (see Figure 5.37)
- identify the specific [DataPrototype](#) that implements a shared axis in the context of a specific access to the enclosing curve (see Figure 5.38)

Figure 5.37 depicts the usage of [SwAxisGrouped](#) for the definition of an [ApplicationPrimitiveDataType](#) that implements a curve.

It is worth noticing that [ApplicationPrimitiveDataType](#) appears in two different roles in the diagram:

- The definition of the [ApplicationPrimitiveDataType](#) of the curve itself is depicted in the upper part of the diagram.
- The selection of the [ApplicationPrimitiveDataType](#) that represents the grouped axis is depicted in the lower part of the diagram.

The modeling depicted in Figure 5.37 can be further clarified by an example model that is reduced to the minimum content to exemplify the use case. Here, the role of the usage of [SwAxisGrouped](#) is to define the data type of the curve itself and of a shared axis:

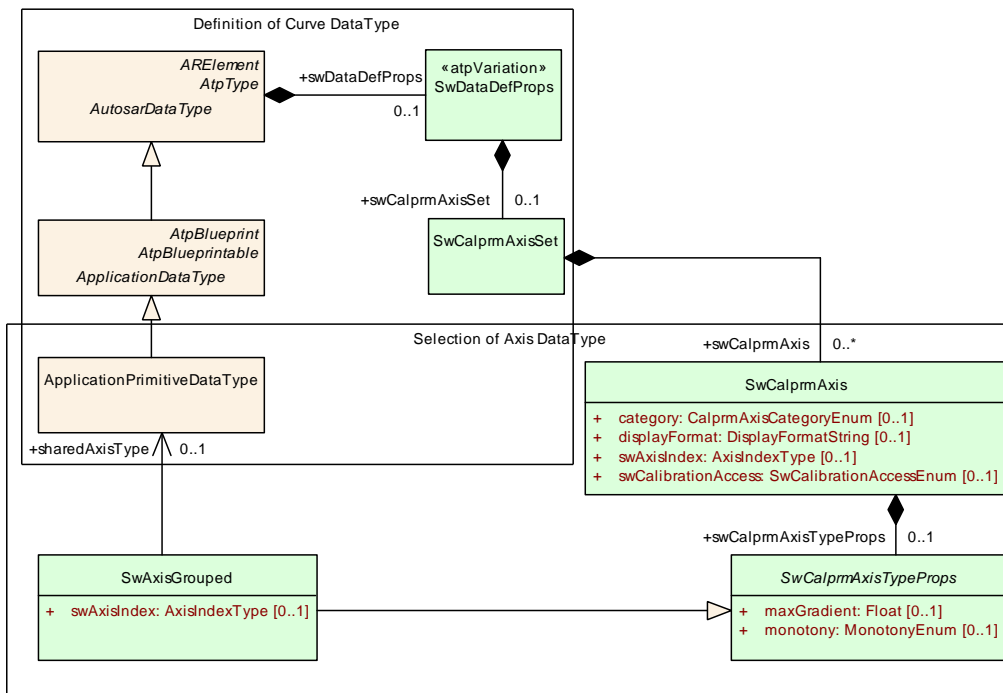


Figure 5.37: Definition of an **ApplicationPrimitiveDataType** that implements a curve

**Listing 5.10: Usage of **SwAxisGrouped** to define the data type of a shared axis**

```

<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>MyAppType</SHORT-NAME>
  <CATEGORY>CURVE</CATEGORY>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALPRM-AXIS-SET>
          <SW-CALPRM-AXIS>
            <SW-AXIS-GROUPED>
              <SHARED-AXIS-TYPE-REF
                DEST="APPLICATION-PRIMITIVE-DATA-TYPE"/>ApplicationDataTypes/
                MyAxisDataType</SHARED-AXIS-TYPE-REF>
            </SW-AXIS-GROUPED>
          </SW-CALPRM-AXIS>
        </SW-CALPRM-AXIS-SET>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-DATA-TYPE>
<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>MyAxisDataType</SHORT-NAME>
</APPLICATION-PRIMITIVE-DATA-TYPE>
    
```

Figure 5.38 depicts the usage of **SwAxisGrouped** in the context of a **Parameter-Access** (which, in the case of a curve, typically results in the call of an interpolation method for the respective curve and the specific axis that is supposed to be used in the interpolation).

Also in this case, the diagram depicts two distinct (yet related) aspects:

- In the upper part of the diagram, the definition of the access to the `DataPrototype` that represents the specific curve inside the enclosing `SwComponentPrototype` is modeled by means of the `ParameterAccess.accessedParameter`.
- In the lower part of the diagram, the identification of the `DataPrototype` that represents the grouped axis inside the enclosing `SwComponentPrototype` is modeled by means of the `SwAxisGrouped.swCalprmRef.arParameter`.

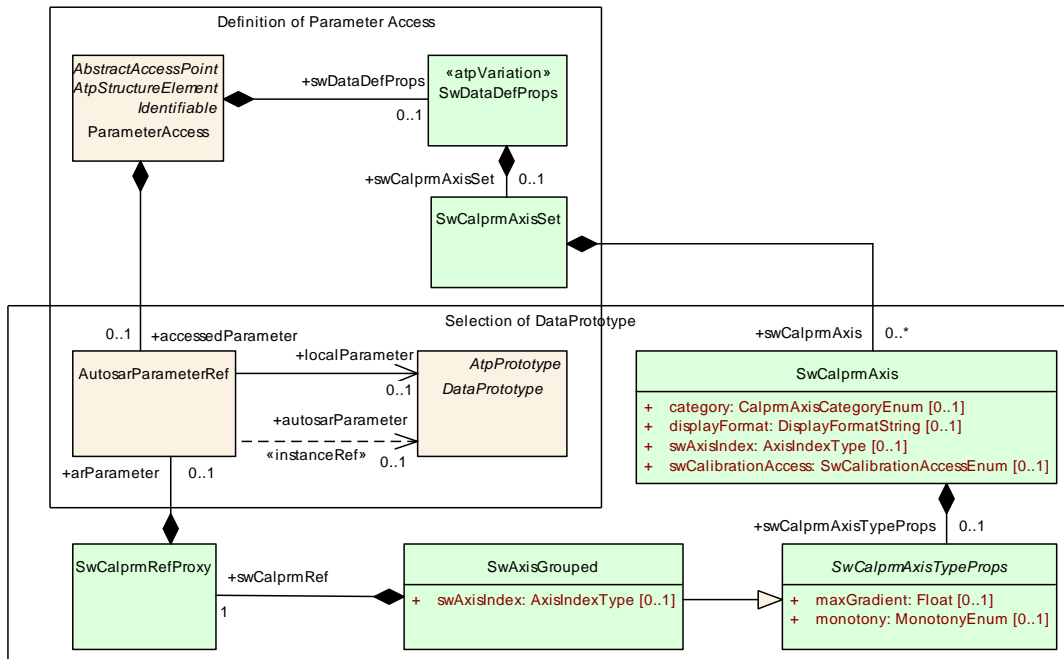


Figure 5.38: Usage of `SwAxisGrouped` in the context of a `ParameterAccess`

The following bare-bones model listing exemplifies the usage of `SwAxisGrouped` in the context of a `ParameterAccess` where a specific local parameter is identified to take the role of the shared axis in the specific access to the curve.

Listing 5.11: Usage of `SwAxisGrouped` in the context of a `ParameterAccess`

```

<APPLICATION-SW-COMPONENT-TYPE>
  <SHORT-NAME>A</SHORT-NAME>
  <INTERNAL-BEHAVIORS>
    <SWC-INTERNAL-BEHAVIOR>
      <SHORT-NAME>B</SHORT-NAME>
      <PER-INSTANCE-PARAMETERS>
        <PARAMETER-DATA-PROTOTYPE>
          <SHORT-NAME>Axis</SHORT-NAME>
        </PARAMETER-DATA-PROTOTYPE>
        <PARAMETER-DATA-PROTOTYPE>
          <SHORT-NAME>Curve</SHORT-NAME>
        </PARAMETER-DATA-PROTOTYPE>
      </PER-INSTANCE-PARAMETERS>
      <RUNNABLES>
        <RUNNABLE-ENTITY>
          <SHORT-NAME>MyRunnable</SHORT-NAME>
        </RUNNABLE-ENTITY>
      </RUNNABLES>
    </SWC-INTERNAL-BEHAVIOR>
  </INTERNAL-BEHAVIORS>
  <PARAMETER-ACCESSES>

```

```

<PARAMETER-ACCESS>
  <SHORT-NAME>MyParamAccess</SHORT-NAME>
  <ACCESSED-PARAMETER>
    <LOCAL-PARAMETER-REF DEST="AUTOSAR-DATA-PROTOTYPE">/P/A/B/
      Curve</LOCAL-PARAMETER-REF>
  </ACCESSED-PARAMETER>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALPRM-AXIS-SET>
          <SW-CALPRM-AXIS>
            <SW-AXIS-GROUPED>
              <AR-PARAMETER>
                <LOCAL-PARAMETER-REF DEST="AUTOSAR-DATA-
                  PROTOTYPE">/P/A/B/Axis</LOCAL-PARAMETER-REF>
              </AR-PARAMETER>
            </SW-AXIS-GROUPED>
          </SW-CALPRM-AXIS>
        </SW-CALPRM-AXIS-SET>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</PARAMETER-ACCESS>
</PARAMETER-ACCESSSS>
</RUNNABLE-ENTITY>
</RUNNABLES>
</SWC-INTERNAL-BEHAVIOR>
</INTERNAL-BEHAVIORS>
</APPLICATION-SW-COMPONENT-TYPE>

```

#### 5.4.4.2 Specification of fix Axes

In most cases the axes of a curve or map are accessible to a calibration software and it is possible to calibrate axes points and their corresponding values.

There are cases, however, where axes are intentionally declared as fix and where no intention exists to change the properties of the axis ever<sup>17</sup>.

These axes are also known as fix axes. The support for the creation of fix axes in the meta-model is based upon the usage of `SwAxisGeneric` as depicted in Figure 5.34.

**[TPS\_SWCT\_01747] Value of `category` for fix axis** [A fix axis shall be modeled as an `SwCalprmAxis` with attribute `category` set to the value `FIX_AXIS`.]()

**[TPS\_SWCT\_01748] Sub-categories of fix axes** [There are different sub-categories of fix axes:

- Fix axis where the distance between axis points can be computed according to a standardized algorithm.

<sup>17</sup>Typically, a calibration software does not have the ability to manipulate (or even inspect) the axis' properties by inspecting the ECU's memory.

In this case, fix axes of arbitrary length can be described by feeding three arguments defined in the context of the axis description into the axis algorithm.

Consequently, the memory footprint of different fix axis of this `category` is literally identical, independently of the number of axis points.

The following variations exist:

- **Subcategory PAR**, i.e. `category = FIX_AXIS_PAR`: the axis is created out of a *starting value* and a *shift* that creates further axis points as using a power-of-two algorithm. The details can be found in [26].
- **Subcategory PAR\_DIST**, i.e. `category = FIX_AXIS_PAR_DIST`: the axis is created out of a *starting value* and an *offset* that adds further axis points with the distance given by offset. The details can be found in [26].
- Fix axis where the axis points are defined as a list of values directly in the axis definition. This variety boils down to
  - **Subcategory PAR\_LIST**, i.e. `category = FIX_AXIS_PAR_LIST`: the axis is created out of a list of numerical values that represent the axis points. The details can be found in [26].

These values of `category` shall be used for `SwAxisType`.`()`

As mentioned before, the modeling of a fix axis is based upon the definition of the `SwAxisGeneric`. But this statement by itself is not yet sufficient to unambiguously clarify the details of the modeling.

For this purpose, it is necessary to provide further information about the specifics of the roles `SwAxisGeneric.swAxisType` and `SwAxisGeneric.swGenericAxisParam`.

**[TPS\_SWCT\_01749] Semantics of `SwAxisGeneric.swAxisType` in the definition of a fix axis** [The role `SwAxisGeneric.swAxisType` specifies the `category` of the fix axis according to [TPS\_SWCT\_01748].]`()`

<code>category</code> of <code>swAxisType</code>	<code>category</code> of <code>SwGenericAxisParamType</code>	Multiplicity of <code>swGenericAxisParam</code>	Multiplicity of <code>vf</code>
<code>FIX_AXIS_PAR</code>	OFFSET	1	1
	SHIFT	1	1
<code>FIX_AXIS_PAR_DIST</code>	OFFSET	1	1
	DISTANCE	1	1
<code>FIX_AXIS_PAR_LIST</code>	LIST	1	1..*

**Table 5.63: Modeling of `SwAxisGeneric`**

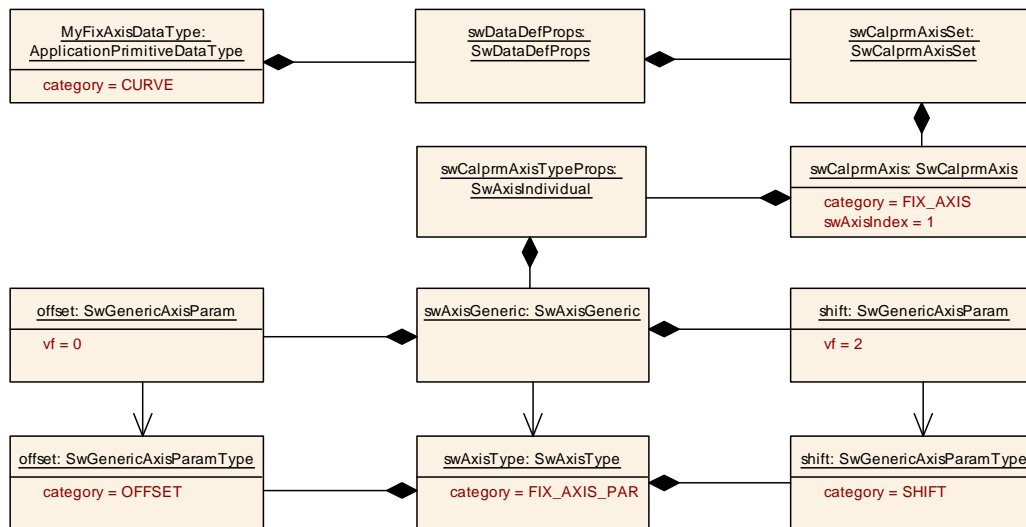
**[constr\_1544] Modeling of `SwAxisGeneric` for the definition of a fix axis** [The standardized values and multiplicities within the model of an `SwAxisGeneric` according to [TPS\_SWCT\_01479] and [TPS\_SWCT\_01480] are documented in Table 5.63.

This rule shall be imposed **at the time when the contract phase generation is executed.**`()`

[TPS\_SWCT\_01750] Semantics of **SwAxisGeneric.swGenericAxisParam** in the definition of a fix axis [The role **SwAxisGeneric.swGenericAxisParam** provides the actual numeric values for the definition of the axis.

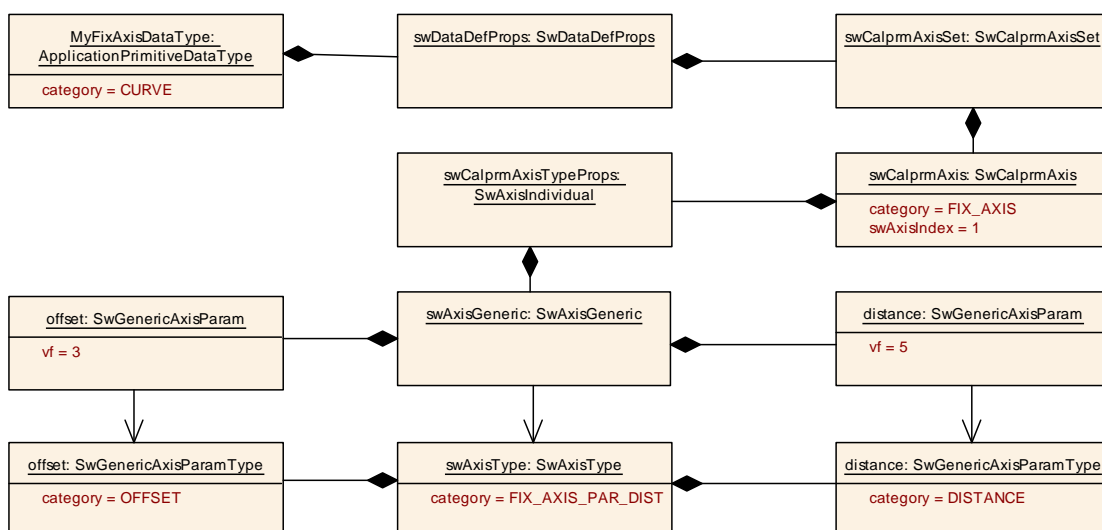
The semantics of a provided numerical value is clarified by the attribute **SwGenericAxisParamType.category** where meta-class **SwGenericAxisParamType** is referenced in the role **swGenericAxisParamType**.]

The modeling of an axis of category **FIX\_AXIS\_PAR** is sketched in the following example model (Figure 5.39).



**Figure 5.39: Modeling of a fix axis of category **FIX\_AXIS\_PAR****

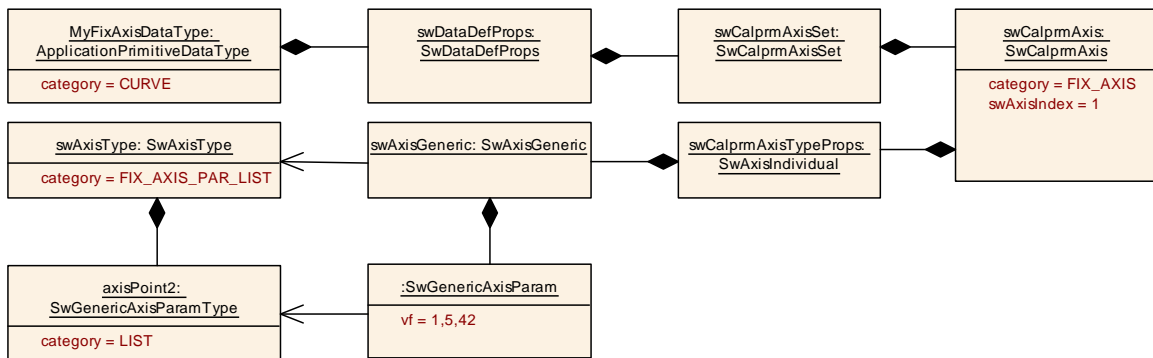
The modeling of an axis of category **FIX\_AXIS\_PAR\_DIST** is sketched in the following example model (Figure 5.40).



**Figure 5.40: Modeling of a fix axis of category **FIX\_AXIS\_PAR\_DIST****

The modeling of an axis of category **FIX\_AXIS\_PAR\_LIST** is sketched in the following example model (Figure 5.41).





**Figure 5.41: Modeling of a fix axis of category `FIX_AXIS_PAR_LIST`**

Please note that the axis points and values of a fix axis are defined in the definition of the fix axis itself and therefore any initial value assigned to a fix axis would be ignored anyway.

This might lead to confusion such that the initial value does not make it into the software. In order to avoid such confusion AUTOSAR **does not support the definition of an initial value for a fix axis.**

This regulation is reflected in the existence of [[constr\\_1545](#)].

**[constr\_1545] No initialization for fix axis** [An [ApplicationValueSpecification](#) taken to initialize an [ApplicationPrimitiveDataType](#) that contains a fix axis shall not contain initial values for the axis index of the fix axis inside the [ApplicationPrimitiveDataType](#) **at any time in the workflow.**]()

Please note that the calibration software may still have access to axis points and values of the fix axis if these properties are specified in an A2L file.

For this purpose [McDataInstance](#) needs to be set up properly. The details are explained in [[6](#)].

### 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 data type of the input values by means of the reference [SwAxisIndividual.inputVariableType](#).

However, the reference [SwAxisIndividual.inputVariableType](#) does not necessarily have to exist.

This leaves the consideration of compatibility between the `DataPrototype(s)` referenced by means of `SwAxisIndividual.swVariableRef` and the actual axis specification to the following attributes:

- `SwAxisIndividual.dataConstr`
- `SwAxisIndividual.compuMethod`
- `SwAxisIndividual.unit`

**[TPS\_SWCT\_01676] Preferred approach to checking the compatibility of input value and axis** [The compatibility in terms of data type between the description of an `SwAxisIndividual` and the `DataPrototype(s)` used as an input variable to the respective interpolation routine shall preferably be checked alternatively between

- the `ApplicationPrimitiveDataType(s)` of `DataPrototype(s)` referenced by means of `SwAxisIndividual.swVariableRef` (the provider in terms of compatibility)
- the `ApplicationPrimitiveDataType` referenced by means of `SwAxisIndividual.inputVariableType` (The requester in terms of compatibility).

For compatibility, the `compuMethod` of `SwAxisIndividual.swVariableRef` and the `ApplicationPrimitiveDataType` referenced by means of `SwAxisIndividual.inputVariableType` shall not be considered.]()

Rationale: in many cases the input variable is defined by a float data type to take benefit from the precision in computations. But the axis data type is an integer data type to save memory. In this situation, a requirement for compatible `compuMethods` would exclude the described scenario.

The implementation of the software-component shall make sure that the float value is properly converted and rescaled to an integer data type compatible to the axis data type.

**[TPS\_SWCT\_01677] Fall-back approach to checking the compatibility of input value and axis** [If the reference `SwAxisIndividual.inputVariableType` does not exist then the compatibility in terms of data type between the description of an `SwAxisIndividual` and the `DataPrototype(s)` used as an input variable to the respective interpolation routine shall be checked on the basis of the following references:

- `SwAxisIndividual.dataConstr`
- `SwAxisIndividual.unit`

respectively

- `SwAxisIndividual.dataConstr`
- `SwAxisIndividual.compuMethod.unit`

against their respective counterparts in the `ApplicationPrimitiveDataTypes` of the `DataPrototype(s)` referenced by means of `SwAxisIndividual.swVariableRef.()`

**[constr\_1420] Existence of `SwAxisIndividual.inputVariableType`** [If the reference `SwAxisIndividual.inputVariableType` does not exist then either:

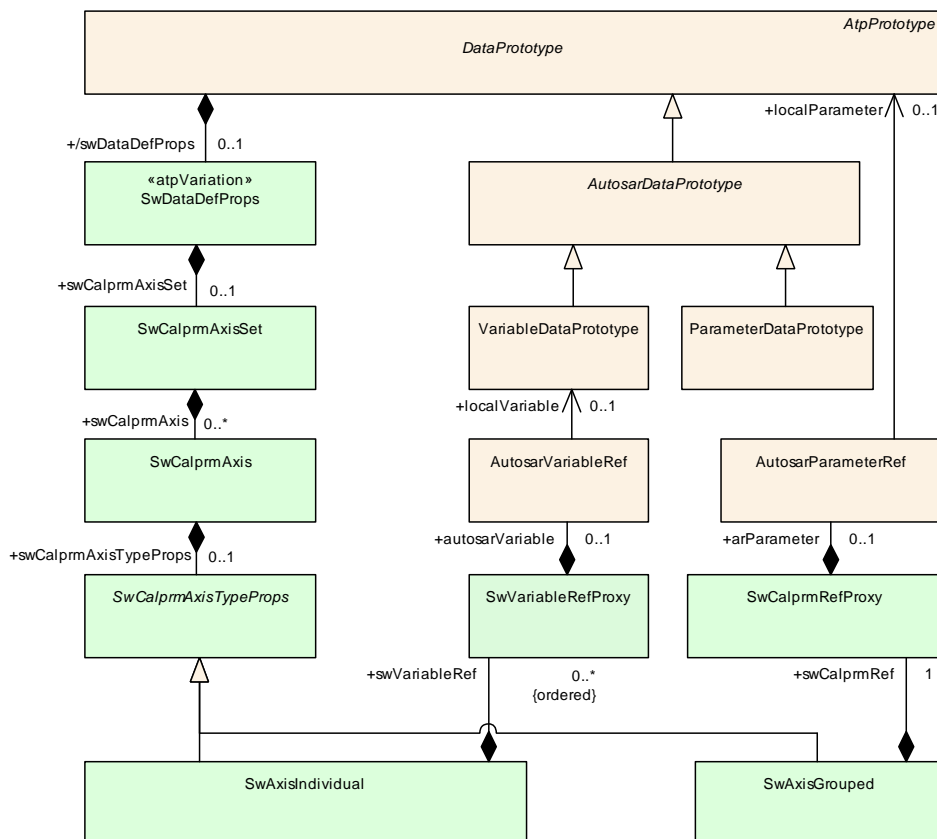
- `SwAxisIndividual.dataConstr`
- `SwAxisIndividual.unit`

or

- `SwAxisIndividual.dataConstr`
- `SwAxisIndividual.compuMethod.unit`

shall exist **at the time when the contract phase generation is executed.**]

The constraint is necessary for the generation of the respective specification of the axis in A2L.



**Figure 5.42: Extended Axis Elements and Input Variable Reference**

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
- a `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 **at any time in the workflow.**]

As shown in Figure 5.42, 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`.

With the means of `ApplicationArrayDataTypes` it's possible to define `DataPrototypes` holding an n-dimensional array of Compound Primitive Data Types of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, or `RES_AXIS`.

For those `DataPrototypes` input values for the axes should be described to enable a display of the working point in the MCD system.

Thereby, typically the whole array of the contained axes is either associated with an array of variables or with a single value. In the case of arrays typically the n-th axis is combined with the n-th input value.

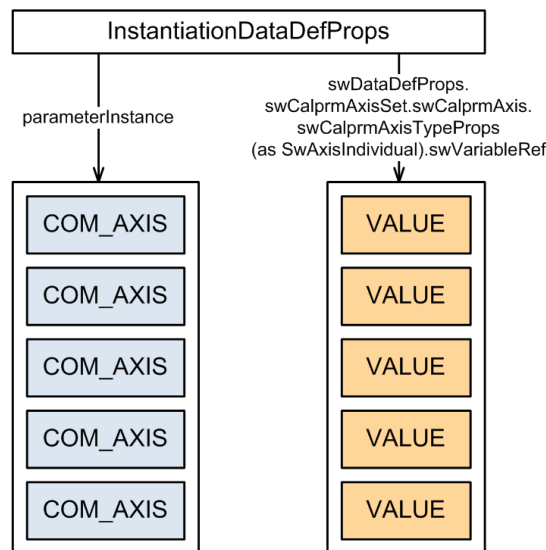
**[`constr_1425`] Definition of `swCalprmAxisSet.swCalprmAxis` / `SwAxisIndividual.swVariableRef` depending on the capabilities of the data type** [The

definition of a `swCalprmAxisSet.swCalprmAxis / SwAxisIndividual.swVariableRef` in the context of an `InstantiationDataDefProps` or a `ParameterAccess` is only supported for a `DataPrototype` of category `ARRAY` if the data type of the `ApplicationArrayElement` also supports the specification of a `swCalprmAxisSet.swCalprmAxis / SwAxisIndividual.swVariableRef` according to [\[constr\\_1289\]](#).

Thereby, multiple `ApplicationArrayDataTypes` might be nested to express multiple array dimensions. This rule shall be imposed **at the time when the contract phase generation is executed.** `()`

**[TPS\_SWCT\_01683] Specification of an array of input variable for an array of axes** [For `DataPrototypes` typed by an array of elements of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, or `RES_AXIS` the applied `InstantiationDataDefProps` or `ParameterAccess` may reference a `VariableDataPrototype` typed by an `ApplicationArrayDataType` with the means of `SwAxisIndividual.swVariableRef.autosarVariable`.

This expresses the semantic that the  $n^{th}$  element in the axis array uses the  $n^{th}$  value in the input variable array for the specific `SwAxisGrouped.swAxisIndex`. `()`



**Figure 5.43: The  $n^{th}$  `COM_AXIS` in the array of `COM_AXIS`s uses the  $n^{th}$  `VALUE` in the array of `VALUE`s as working point.**

Please note that in this case the two associated arrays need to have the same number of dimensions and sizes of the dimensions.

**[constr\_1426] Consistency of array sizes for axes and input variable array** [The number of array dimension defined by `ApplicationArrayDataTypes` and the values of the `maxNumberOfElements` attributes for the array of elements of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, or `RES_AXIS` shall be **identical** to the number of array dimension and according value of the `maxNumberOfElements` of the `VariableDataPrototype` referenced by `SwAxisIndividual.swVariableRef.autosarVariable`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

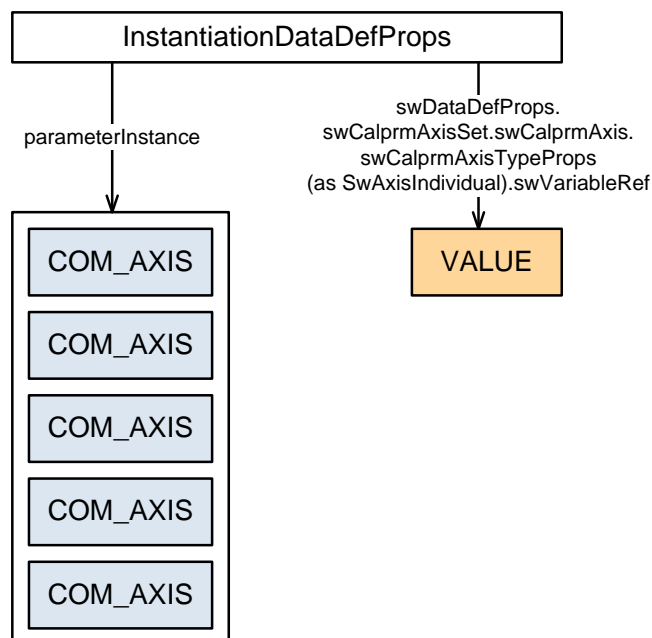


Figure 5.44: Each **COM\_AXIS** in the array of **COM\_AXIS**s uses the identical **VALUE** as working point.

**[TPS\_SWCT\_01684] Specification of a single input variable for an array of axes**  
 [For *DataPrototypes* typed by an array of elements of *category* CURVE, MAP, CUBOID, CUBE\_4, CUBE\_5, COM\_AXIS, or RES\_AXIS the applied *InstantiationDataDefProps* or *ParameterAccess* may reference a *VariableDataPrototype* typed by an *ApplicationPrimitiveDataType* with the means of *SwAxisIndividual.swVariableRef.autosarVariable*.

This expresses the semantic that each element in the axis array uses the identical input variable for the specific *SwAxisGrouped.swAxisIndex*.]()

#### 5.4.6 Setting a Group Axis

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*.

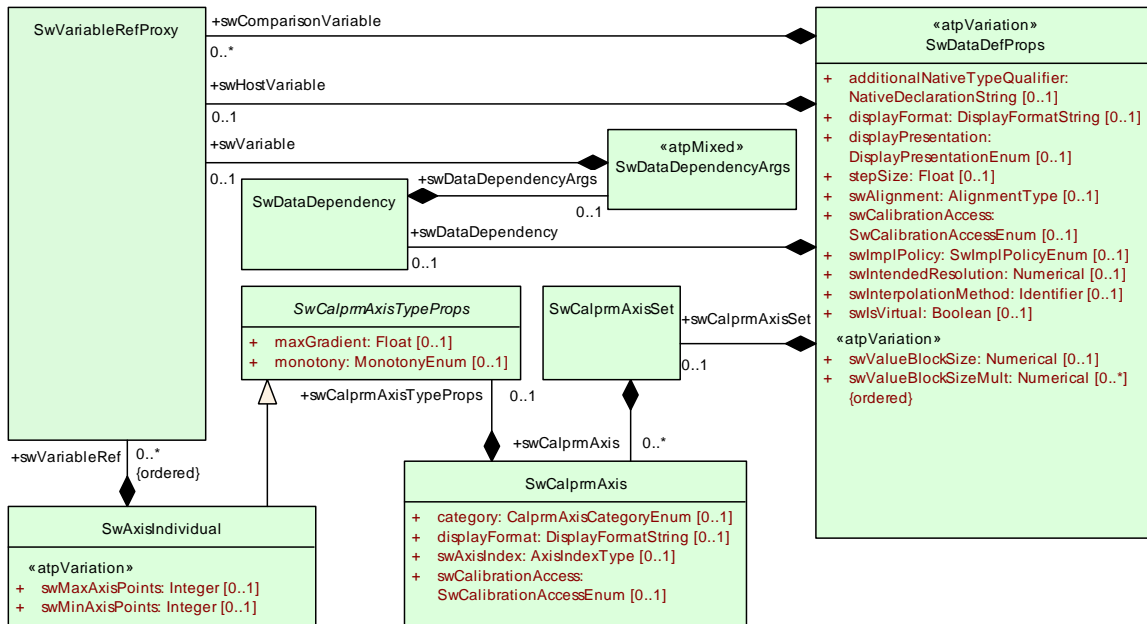
AUTOSAR applies a similar proxy approach for parameters as for the variables. Therefore, an *SwCalprmRefProxy* has been introduced in MSRSW, and is aggregated by the *SwAxisGrouped* element.

The *SwCalprmRefProxy* aggregates an *AutosarParameterRef* providing an association to a *ParameterDataPrototype*, representing a curve with an axis. When

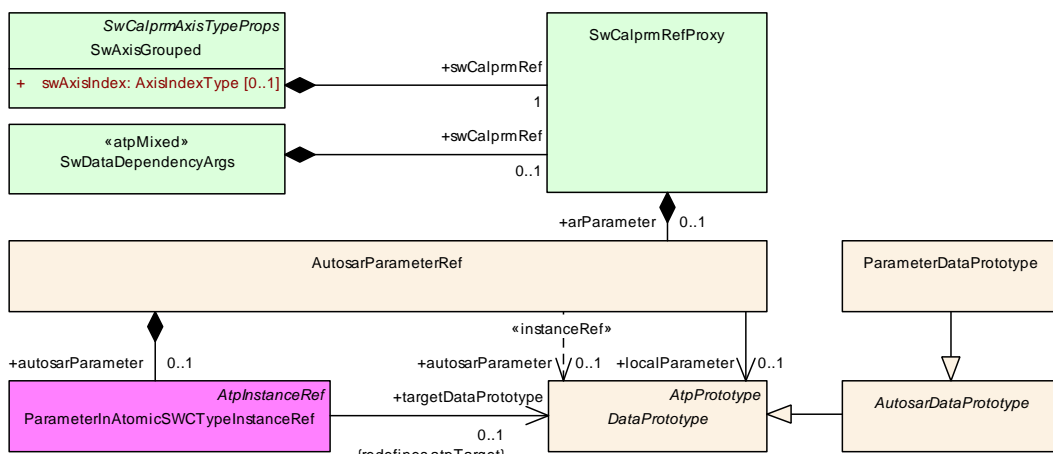
defining the data-type of a parameter, the type of the shared axis is defined in `sharedAxisType`.

**[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` at the time when the contract phase generation is executed.]()

The AUTOSAR-style is shown in the upper left part of Figure 5.42, while in the upper middle the MSRSW style is shown, referencing the `SwCalprm`.



**Figure 5.45: Applying Proxy Variable Reference Mechanism**



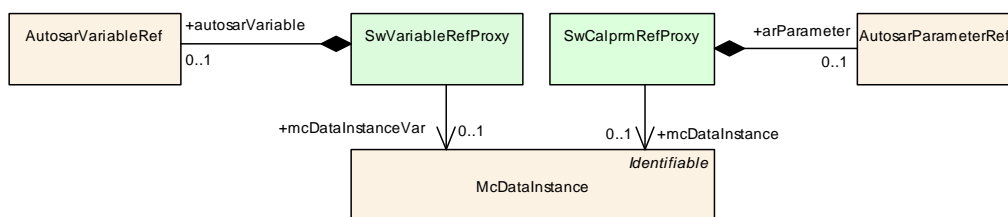
**Figure 5.46: Applying Proxy Parameter Reference Mechanism**

<b>Class</b>	<b>SwCalprmRefProxy</b>			
<b>Package</b>	M2::MSR::DataDictionary::DatadictionaryProxies			
<b>Note</b>	Wrapper class for different kinds of references to a calibration parameter.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arParameter	<a href="#">AutosarParameterRef</a>	0..1	aggr	This represents a Parameter within AUTOSAR. Note that the Datatype of the referenced ParameterDataPrototype shall be an ApplicationDataType of category VALUE.
mcDataInstance	<a href="#">McDataInstance</a>	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 5.64: SwCalprmRefProxy**

<b>Class</b>	<b>SwVariableRefProxy</b>			
<b>Package</b>	M2::MSR::DataDictionary::DatadictionaryProxies			
<b>Note</b>	Proxy class for several kinds of references to a variable.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
autosarVariable	<a href="#">AutosarVariableRef</a>	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
mcDataInstance Var	<a href="#">McDataInstance</a>	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 5.65: SwVariableRefProxy**



**Figure 5.47: Proxy reference classes**

The basic patterns for referencing [DataPrototypes](#) are explained in section 5.3.3. 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 5.43.

With the means of `ApplicationArrayDataTypes` its possible to define `DataPrototypes` holding an n-dimensional array of Compound Primitive Data Types of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, and `CUBE_5`.

For those `DataPrototypes`, group axis/axes needs to be defined in case `SwAxisIndividuals` are not used for all `SwCalprmAxis` definitions.

Thereby, typically the whole array of elements of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, and `CUBE_5` is either associated with an array of group axes or alternatively with a single group axis.

In the case of arrays typically the n<sup>th</sup> `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, and `CUBE_5` is combined with the n<sup>th</sup> `COM_AXIS` or `RES_AXIS`.

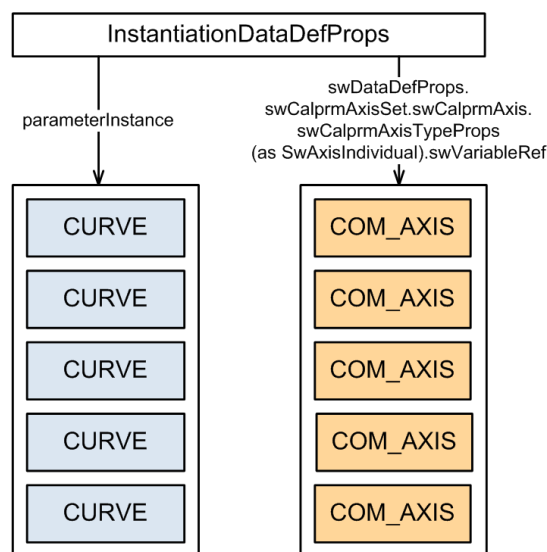


Figure 5.48: The n<sup>th</sup> `CURVE` in the array of `CURVES` relates to the n<sup>th</sup> `COM_AXIS` in the array of `COM_AXISs`

**[constr\_1427] Definition of `swCalprmAxisSet.swCalprmAxis` / `SwAxisGrouped.swCalprmRef` depending on the capabilities of the data type** [The definition of a `swCalprmAxisSet.swCalprmAxis` / `SwAxisGrouped.swCalprmRef` in the context of an `InstantiationDataDefProps` or a `ParameterAccess` is only supported for a `DataPrototype` of category `ARRAY` if the data type of the `ApplicationArrayElement` also supports the specification of a `swCalprmAxisSet.swCalprmAxis` / `SwAxisGrouped.swCalprmRef` according to [constr\_1289].

Thereby, multiple `ApplicationArrayDataTypes` might be nested to express multiple array dimensions. This rule shall be imposed **at the time when the contract phase generation is executed.**]()

[TPS\_SWCT\_01685] Specification of an array of group axes for an array of **category CURVE, MAP, CUBOID, CUBE\_4, or CUBE\_5** [For DataPrototypes typed by an array of elements of **category CURVE, MAP, CUBOID, CUBE\_4, or CUBE\_5** the applied **InstantiationDataDefProps** or **ParameterAccess** may reference a **DataPrototype** typed by an **ApplicationArrayDataType** with the means of **SwAxisGrouped.swCalprmRef.arParameter**.

This expresses the semantic that the  $n^{\text{th}}$  element in the **CURVE, MAP, CUBOID, CUBE\_4, or CUBE\_5** array uses the  $n^{\text{th}}$  group axis in the **COM\_AXIS** or **RES\_AXIS** array for the specific **SwAxisGrouped.swAxisIndex**.]()

Please note that in this case the two associated arrays needs to have same number of dimensions and sizes of the dimensions.

[constr\_1428] Consistency of array sizes for arrays of elements of **category CURVE, MAP, CUBOID, CUBE\_4, or CUBE\_5** arrays and used group axes arrays [The number of array dimension defined by **ApplicationArrayDataTypes** and the values of attribute **maxNumberOfElements** attributes for the array of elements of **category CURVE, MAP, CUBOID, CUBE\_4, or CUBE\_5** needs to be identical to the number of array dimension and according value of the **maxNumberOfElements** of the **DataPrototype** referenced by **SwAxisGrouped.swCalprmRef.arParameter** **at the time when the contract phase generation is executed**.]()

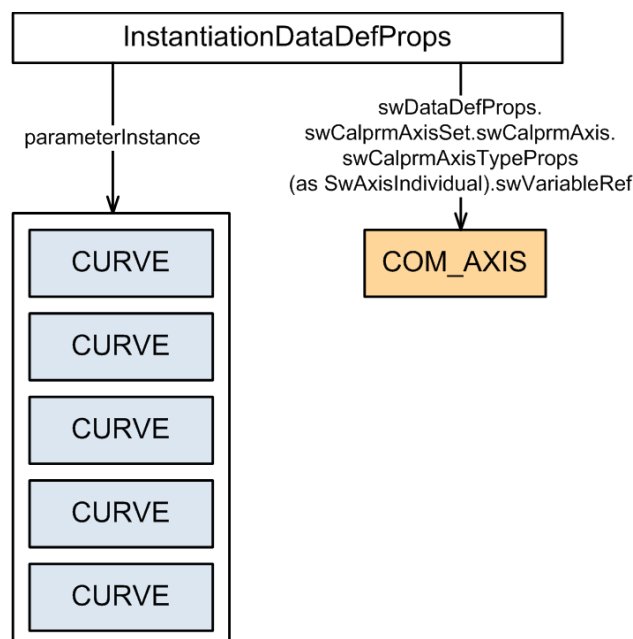


Figure 5.49: Each **MAP** in the array of **CURVES** uses the identical **COM\_AXIS**

[TPS\_SWCT\_01686] Specification of a single group axis for an array of elements of **category CURVE, MAP, CUBOID, CUBE\_4, or CUBE\_5** [For **DataPrototypes** typed by an array of elements of **category CURVE, MAP, CUBOID, CUBE\_4, or CUBE\_5** the applied **InstantiationDataDefProps** or **ParameterAccess** may reference a **DataPrototype** typed by a **ApplicationPrimitiveDataTypes** of category

COM\_AXIS or RES\_AXIS with the means of `SwAxisGrouped.swCalprmRef.arParameter`.

This expresses the semantic that each element in the CURVE, MAP, CUBOID, CUBE\_4, or CUBE\_5 array uses the identical COM\_AXIS or RES\_AXIS for the specific `SwAxisGrouped.swAxisIndex.()`

## 5.4.7 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 parameters A and B. When adjusting parameter 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 5.12.

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 5.12: 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>
</PER-INSTANCE-PARAMETERS>
    
```

```

        </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>

```

<b>Class</b>	<b>SwDataDependency</b>			
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties			
<b>Note</b>	<p>This element describes the interdependencies of data objects, e.g. variables and parameters.</p> <p>Use cases:</p> <ul style="list-style-type: none"> <li>Calculate the value of a calibration parameter (by the MCD system) from the value(s) of other calibration parameters.</li> <li>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).</li> </ul>			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swDataDependencyArgs	<a href="#">SwDataDependencyArgs</a>	0..1	aggr	<p>Specifies the arguments used in the data dependency. Note that this is 0..1 since the aggregated class is a container (atpMixed).</p> <p><b>Tags:</b>xml.sequenceOffset=40</p>
swDataDependencyFormula	CompuGenericMath	0..1	aggr	<p>This element describes the formula with which the dependencies between the participating objects are defined.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>

**Table 5.66: SwDataDependency**

<b>Class</b>	<<atpMixed>> <b>SwDataDependencyArgs</b>			
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties			
<b>Note</b>	This element specifies the elements used in a SwDataDependency.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swCalprmRef	<a href="#">SwCalprmRefProxy</a>	0..1	aggr	<p>Specifies a calibration parameter as an input argument to the dependency.</p> <p><b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=60 xml.typeElement=false xml.typeWrapperElement=false</p>
swVariable	<a href="#">SwVariableRefProxy</a>	0..1	aggr	<p>Specifies a variable as an input argument to the dependency.</p> <p><b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false</p>

**Table 5.67: SwDataDependencyArgs**

### 5.4.8 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.

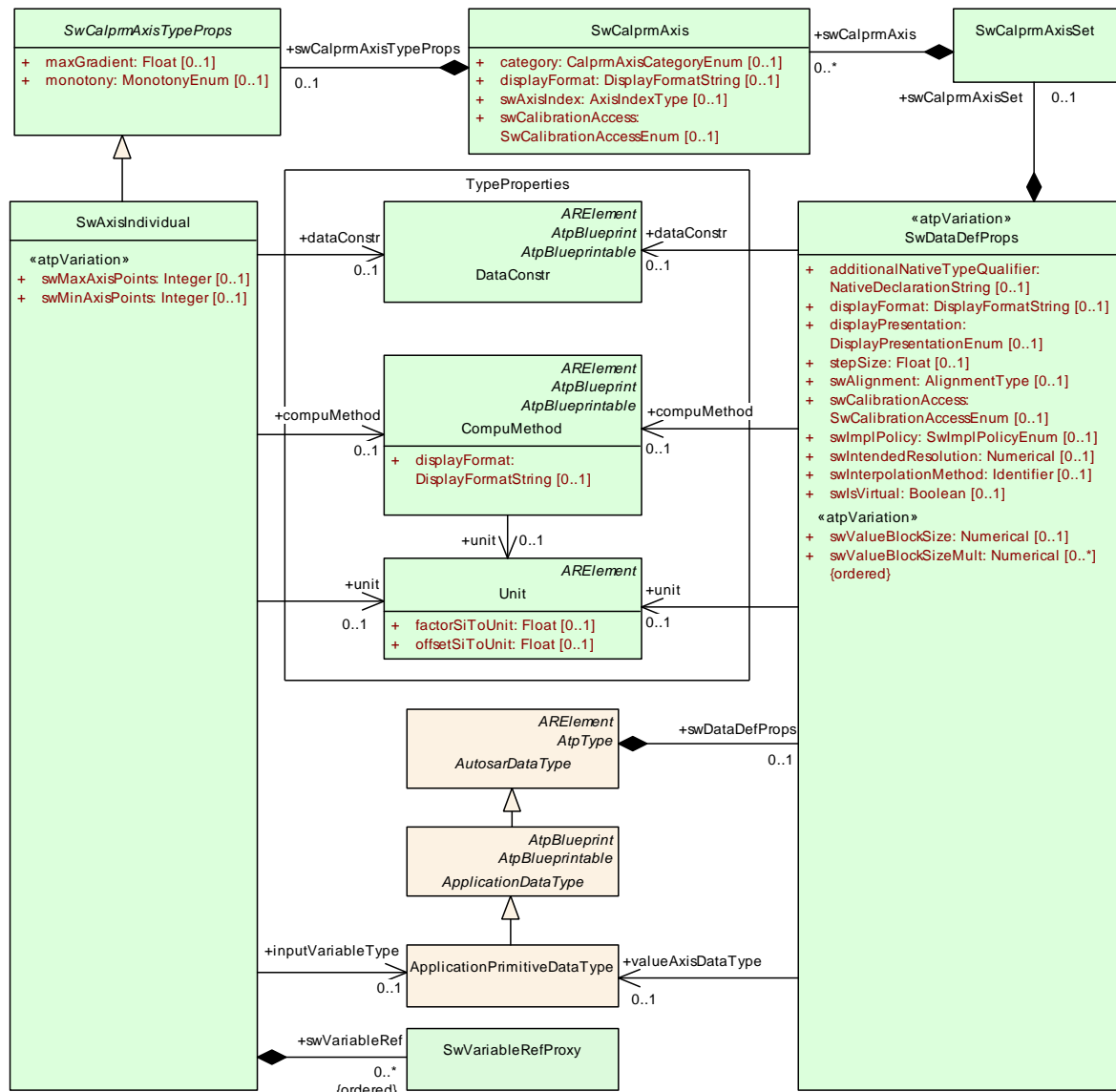


Figure 5.50: Various Attributes in the Context of `SwDataDefProps`

Figure 5.50 illustrates the fact that some attributes in `SwDataDefProps` can also be expressed in sub-elements 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` or `SwAxisIndividual.inputVariableType`.

- `SwAxisIndividual.inputVariableType` wins over `SwAxisIndividual.compuMethod`, `SwAxisIndividual.unit`, but **not** over `SwAxisIndividual.dataConstr`.

]()

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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

In particular, **[constr\_2550]** reflects the fact that a `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 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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

- **[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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

[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`

]()

Please note that the attribute `SwAxisIndividual.inputVariableType.swDataDefProps.dataConstr` represents the input value, not the axis itself. For this reason, there is no specific constraint defined that the `dataConstr` needs to fulfill.

- **[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 5.68.]()

The interpretation of table 5.68 is that 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.

outer	inner	result
<code>notAccessible</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 5.68: Precedence of `swCalibrationAccess` along structure hierarchies**

## 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](#).

<b>Class</b>	<b>CompuMethod</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	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.  <b>Tags:</b> atp.recommendedPackage=CompuMethods			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuInternalToPhys	<a href="#">Compu</a>	0..1	aggr	This specifies the computation from internal values to physical values.  <b>Tags:</b> xml.sequenceOffset=80
compuPhysToInternal	<a href="#">Compu</a>	0..1	aggr	This represents the computation from physical values to the internal values.  <b>Tags:</b> xml.sequenceOffset=90
displayFormat	<a href="#">DisplayFormatString</a>	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools.  <b>Tags:</b> xml.sequenceOffset=20
unit	<a href="#">Unit</a>	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies.  <b>Tags:</b> xml.sequenceOffset=30

**Table 5.69: 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.

Some [category](#)s of [CompuMethod](#) cannot be successfully converted to A2L [26] because A2L does not provide an equivalent semantics that comes close to the respective AUTOSAR semantics.

A prominent example for such a case is a [CompuMethod](#) of category [SCALE\\_LINEAR\\_AND\\_TEXTTABLE](#) that actually has more than one linear interval and a texttable part.

**[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` at any time in the workflow. ]()

**[TPS\_SWCT\_01277] Computation methods are used for the conversion of internal values into their physical representation and vice versa** [CompuMethods 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.

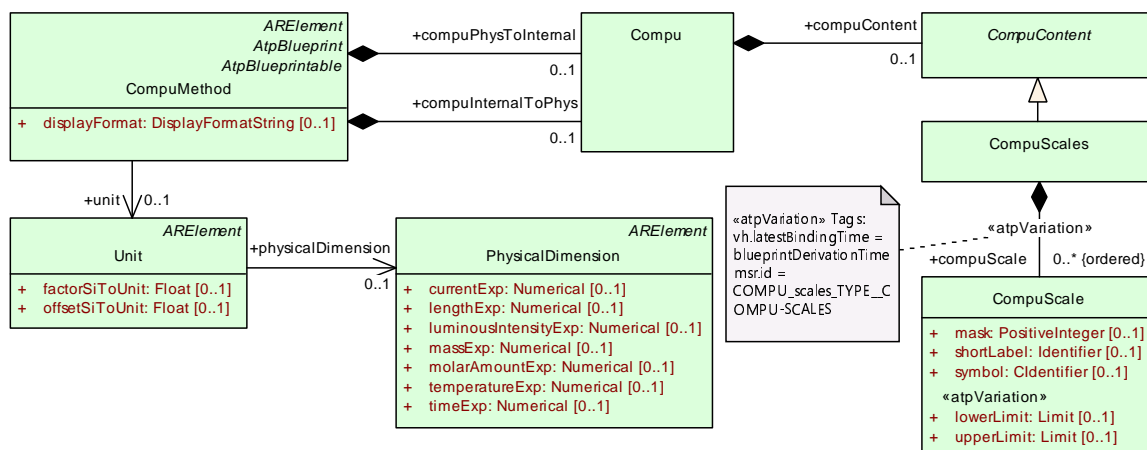
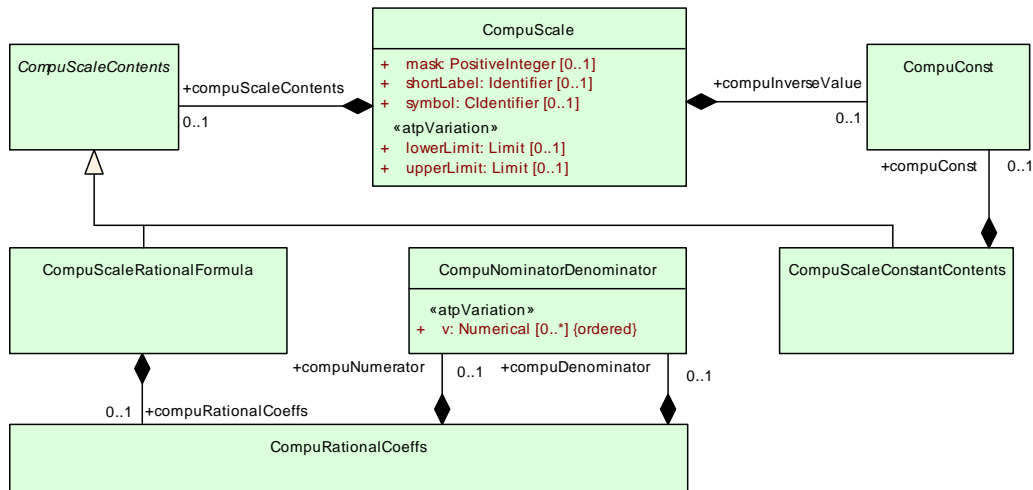


Figure 5.51: A `CompuMethod` and its attributes define data semantics

**[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.] ()

Actually, the preferred conversion direction depends on the use case.



**Figure 5.52: A `CompuScale` and its attributes define data semantics**

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 directly, an inverse value can be provided explicitly for the function or for each of its parts respectively.

**[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 **at the time when the contract phase generation is executed.**] ()

**[TPS\_SWCT\_01280] `CompuMethod` applied to values outside of its limits** [If a `CompuMethod` is applied to values outside 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, it shall refer to a `unit` unless the `CompuMethod`'s `category` is one of `TEXTTABLE`, `BITFIELD_TEXTTABLE`, or `IDENTICAL`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

[**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 5.51 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** [In a bound model, the intervals (i.e. determined by attributes `CompuScale.lowerLimit` and `CompuScale.upperLimit`) defined by `CompuScales` used in the context of a given `CompuMethod` of all values of `category` except `BITFIELD_TEXTTABLE` shall **not** overlap.

For `CompuMethods` of `category` `BITFIELD_TEXTTABLE`, the combination of the interval created by attributes `CompuScale.upperLimit`, `CompuScale.lowerLimit` and `CompuScale.mask` shall be unique in the context of the enclosing `CompuMethod`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

The possible values of `CompuMethod.category` are listed in Table 5.80.

[**TPS\_SWCT\_01667**] **Avoidance of overlapping of directly adjacent intervals within CompuMethods** [Intervals of a given `CompuMethod` may be **located directly adjacent** to each other.

This means that the `upperLimit` of one `CompuScale` has the same numerical value as the `lowerLimit` of another `CompuScale` defined within the context of the `CompuMethod`.

In this case, it is necessary to properly set the attribute `CompuScale.lowerLimit.intervalType` or `CompuScale.upperLimit.intervalType` in order to avoid an overlapping.

Specifically, one of the interval boundaries shall be set to `intervalType.open` in order to avoid an overlapping.]()

- **[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.

Within each `CompuScale` we have the abstract `CompuScaleContents`. To deal with possible ambiguities directly, an inverse value can be provided explicitly for that particular scale (`compuInverseValue`).]()

Please note that limits are explained in more detail in chapter 5.2.4.1.

- 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 (`CompuRationalCoeffs`)
- **[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  $x_0$ , the second *V* is the coefficient for  $x_1$ , 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 **at any time in the workflow**.]()

**[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 Code 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` Code 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 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.

]

**[TPS\_SWCT\_01695] Relation between `ValueSpecification` and the definition of `CompuScales`** [In order to find a match between the content of a `ValueSpecification` and a `CompuScale` the content of the `ValueSpecification` shall be checked against the `CompuScale Value Symbolic Names` according to [TPS\_SWCT\_01696].

If no matching `CompuScale Value Symbolic Names` can be found then the `ValueSpecification` shall be considered unusable in the context of the `CompuMethod` that is subject to [constr\_1146].

**[TPS\_SWCT\_01696] `CompuScale Value Symbolic Name`** [The value of the `CompuScale Value Symbolic Name` of a given `CompuScale` shall be obtained by taking the values of the following attributes according to the following priority (lower values indicate higher priority):

1. Take the value of `symbol` if this attribute exists.
2. Take the value of `vt` if this attribute exists.
3. Take the value of `shortLabel` if it exists.

]

Just to be sure, the (obvious) difference between a `CompuScale Value Symbolic Name` and a `CompuScale Code Symbolic Name` is that the former is not required to pass as a valid C identifier.



**[constr\_1434] CompuScales shall not have identical CompuScale Value Symbolic Names** [In a CompuMethod that is subject to [constr\_1146], no two CompuScales shall have identical CompuScale Value Symbolic Names (according to [TPS\_SWCT\_01696]) **at the time when the contract phase generation is executed.**]()

**[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:

- TEXTTABLE
- SCALE\_LINEAR\_AND\_TEXTTABLE
- SCALE\_RATIONAL\_AND\_TEXTTABLE
- BITFIELD\_TEXTTABLE

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

<b>Class</b>	<b>Compu</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to express one particular computation.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuContent	CompuContent	0..1	aggr	This specifies the details of the computation. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
compuDefault Value	CompuConst	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. <b>Tags:</b> xml.sequenceOffset=70

**Table 5.70: Compu**

<b>Class</b>	<b>CompuContent</b> (abstract)			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This abstract meta-class represents the various definition means of a computation method.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	CompuScales			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 5.71: CompuContent**

<b>Class</b>	<b>CompuScale</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to specify one segment of a segmented computation method.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuInverse Value	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. <b>Tags:</b> xml.sequenceOffset=60
compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. <b>Tags:</b> xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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. <b>Tags:</b> xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. <b>Tags:</b> xml.sequenceOffset=20
symbol	CIdentifier	0..1	attr	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. <b>Tags:</b> xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

**Table 5.72: CompuScale**

<b>Class</b>	<b>CompuScales</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to stepwise express a computation method.			
<b>Base</b>	ARObject, CompuContent			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false

**Table 5.73: CompuScales**

<b>Class</b>	<b>CompuScaleContents</b> (abstract)			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This abstract meta-class represents the content of one particular scale.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	CompuScaleConstantContents, CompuScaleRationalFormula			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 5.74: CompuScaleContents**

<b>Class</b>	<b>CompuRationalCoeffs</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to express a rational function by specifying the coefficients of nominator and denominator.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compu Denominator	CompuNominator Denominator	0..1	aggr	This is the denominator of the expression. <b>Tags:</b> xml.sequenceOffset=30
compu Numerator	CompuNominator Denominator	0..1	aggr	This is the numerator of the rational expression. <b>Tags:</b> xml.sequenceOffset=20

**Table 5.75: CompuRationalCoeffs**

**[constr\_1913] Existence of attribute [CompuRationalCoeffs.compuDenominator](#)** [For each [CompuRationalCoeffs](#), the attribute [compuDenominator](#) shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1914] Existence of attribute [CompuRationalCoeffs.compuNumerator](#)** [For each [CompuRationalCoeffs](#), the attribute [compuNumerator](#) shall exist **at the time when the contract phase generation is executed.**]()

<b>Class</b>	<b>CompuConst</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the fact that the value of a computation method scale is constant.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuConst Content ContentType	CompuConstContent	0..1	aggr	This is the actual content of the constant compu method scale.  <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=10 xml.typeElement=false xml.typeWrapperElement=false

**Table 5.76: CompuConst**

<b>Class</b>	<b>CompuScaleRationalFormula</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the fact that the computation in this scale is represented as rational term.			
<b>Base</b>	ARObject, <a href="#">CompuScaleContents</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuRational Coeffs	<a href="#">CompuRationalCoeffs</a>	0..1	aggr	This specifies the coefficients of the rational formula.  <b>Tags:</b> xml.sequenceOffset=110

**Table 5.77: CompuScaleRationalFormula**

<b>Class</b>	<b>CompuScaleConstantContents</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the fact that a particular scale of the computation method is constant.			
<b>Base</b>	ARObject, <a href="#">CompuScaleContents</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuConst	<a href="#">CompuConst</a>	0..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.  <b>Tags:</b> xml.sequenceOffset=90

**Table 5.78: CompuScaleConstantContents**

<b>Class</b>	<b>CompuNominatorDenominator</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This class represents the ability to express a polynomial either as Nominator or as Denominator.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	CompuNominatorDenominator			
v (ordered)	Numerical	*	attr	this is the list of polynomial factors. Note that the first v <sub>f</sub> represents the power=0. The polynomial is v <sub>[0]</sub> * x <sup>0</sup> + v <sub>[1]</sub> * x <sup>1</sup> ...  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

**Table 5.79: 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* [27].

Table 5.80 contains a definition of possible values for the attribute `category`.

ASAM Category	Meaning	Specific properties
<b>IDENTICAL</b>	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>physConstrs</code> and <code>internalConstrs</code> are optional. This is the simplest type of a <code>CompuMethod</code> .
<b>LINEAR</b>	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> .
<b>SCALE_LINEAR</b>	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.
<b>SCALE_LINEAR_AND_TEXTTABLE</b>	Used for piecewise definition of one or more 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.





ASAM Category	Meaning	Specific properties
<b>RAT_FUNC</b>	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.
<b>SCALE_RAT_FUNC</b>	Used for piecewise defined rational conversion.	
<b>SCALE_RATIONAL_AND_TEXTTABLE</b>	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.
<b>TEXTTABLE</b>	The type <code>TEXTTABLE</code> is used for transformations of the internal value into textual elements.	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.  [constr_1134] applies!
<b>TAB_NOINTP</b>	Similar to <code>TEXTTABLE</code> , but for numerical values.	The values per scale are defined in <code>CompuConst</code> .
<b>BITFIELD_TEXTTABLE</b>	Similar to <code>TEXTTABLE</code> but for bit fields.	<code>BITFIELD_TEXTTABLE</code> is derived from <code>TEXTTABLE</code> . The main difference is that <code>TEXTTABLE</code> results to a single value while <code>BITFIELD_TEXTTABLE</code> results to a concatenated value set.  In difference to all the other computational methods <b>every</b> <code>CompuScale</code> will be applied including the bit mask specified in <code>mask</code> . Therefore it is allowed for this type of <code>CompuMethod</code> , that <code>CompuScales</code> 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 <code>CompuScale</code> elements.  [constr_1135] applies!

**Table 5.80: ASAM compuMethod**

**[constr\_1134] Allowed structure of `TEXTTABLE`** [The existence of `physConstrs` is not allowed and `compuInternalToPhys` shall exist with `compuScales` consisting of `upperLimit` and `lowerLimit` **at any time in the workflow.**]()

**[constr\_1135] Limit of `vt` in `BITFIELD_TEXTTABLE`** [The separator for splitting the string representing the value is “|” and is therefore forbidden to appear in `vt` **at any time in the workflow.**]()

### 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 5.81.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

	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
<b>Attributes of <code>CompuMethod</code></b>										
<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(1)	D(1)	D(2)	D(2)	N/A	N/A	N/A	D(2,3)	N/A
<b>Attributes of meta-classes related to <code>CompuMethod</code></b>										
<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.compuNumerator</code>	N/A	D/2v	D/2v	D	D	N/A	N/A	D/2v	D	N/A

**Table 5.81: Allowed Attributes vs. `category` for `CompuMethods`**

For clarification, the first two rows of Table 5.81 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 5.81.

The following legend applies to the cells in table 5.81:

**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 5.81:

- (1) In this case **either** `compuPhysToInternal` **or** `compuInternalToPhys` shall be defined.
- (2) In this case both `compuPhysToInternal` and `compuInternalToPhys` shall be defined 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 CompuMethod and AutosarDataType

This chapter clarifies the applicability of `CompuMethod` for the relevant concrete sub-classes of `AutosarDataType`.

#### 5.5.1.3.1 CompuMethod and ApplicationDataType

For `ApplicationDataType`, there are (see Table 5.7) a number values of `category` that allow for the definition of a `ApplicationDataType.swDataDefProps.compuMethod`.

Table 5.82 visualizes the allowed combinations of `ApplicationDataType.category` vs. `CompuMethod.category`.



	IDENTICAL	LINEAR	SCALE_LINEAR	SCALE_LINEAR_AND_TEXTTABLE	RAT_FUNC	SCALE_RATIONAL_AND_TEXTTABLE	TEXTTABLE	TAB_NOINTP	BITFIELD_TEXTTABLE
VALUE	x	x	x	x	x	x	x	x	x
VAL_BLK	x	x	x	x	x	x	x	x	x
BOOLEAN	n/a	n/a	n/a	n/a	n/a	n/a	x	n/a	n/a
CURVE	x	x	x	x	x	x	x	x	x
MAP	x	x	x	x	x	x	x	x	x
CUBOID	x	x	x	x	x	x	x	x	x
CUBE_4	x	x	x	x	x	x	x	x	x
CUBE_5	x	x	x	x	x	x	x	x	x

**Table 5.82: `ApplicationDataType.category` vs. `CompuMethod.category`**

The rows of Table 5.82 represent values of `category` for `ApplicationDataType` that are cleared for the definition of a `CompuMethod` according to Table 5.7.

The columns of Table 5.82 represent values of `category` for `CompuMethod`.

**[constr\_1634] Allowed combinations of `ApplicationDataType.category` vs. `CompuMethod.category`** [The allowed combinations of `ApplicationDataType.category` vs. `CompuMethod.category` at the time when the contract phase generation is executed are described by Table 5.82.]()

### 5.5.1.3.2 `CompuMethod` and `ImplementationDataType`

For `ImplementationDataType`, there are (see Table 5.17) only two values of `category` that allow for the definition of a `ImplementationDataType.swDataDefProps.compuMethod`: `TEXTTABLE` and `BITFIELD_TEXTTABLE`.

**[constr\_1158] Applicable `categories` 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).

This rule shall be imposed at the time when the contract phase generation is executed.]()

The statement made by [constr\_1158] is further visualized by Table 5.83.

	IDENTICAL	LINEAR	SCALE_LINEAR	SCALE_LINEAR_AND_TEXTTABLE	RAT_FUNC	SCALE_RATIONAL_AND_TEXTTABLE	TEXTTABLE	TAB_NOINTP	BITFIELD_TEXTTABLE
VALUE	n/a	n/a	n/a	n/a	n/a	n/a	x	n/a	x
TYPE_REFERENCE	n/a	n/a	n/a	n/a	n/a	n/a	x	n/a	x

**Table 5.83: ImplementationDataType.category vs. CompuMethod.category**

The rows of Table 5.83 represent values of `category` for `ImplementationDataType` that are cleared for the definition of a `CompuMethod` according to Table 5.17.

The columns of Table 5.83 represent values of `category` for `CompuMethod`.

#### 5.5.1.4 Example for Enumeration

The following example illustrates how an enumeration is specified using `CompuMethod`.

**Listing 5.13: 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>
    
```

### 5.5.1.5 Example for Linear Conversion

The following examples illustrate how a linear conversion is specified using `CompuMethod`.

$$F_{[km/h]} = 30_{[km/h]} + 2_{[km/h]} * x$$

Listing 5.14: example for linear `CompuMethod`

```

<COMPU-METHOD>
  <SHORT-NAME>linear</SHORT-NAME>
  <CATEGORY>LINEAR</CATEGORY>
  <UNIT-REF DEST="UNIT">kmph</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>
    
```

### 5.5.1.6 Example for Linear Conversion with texttable

The following example illustrates how a linear conversion with a texttable is specified using `CompuMethod`.

Listing 5.15: example for linear and texttable `CompuMethod`

```

<COMPU-METHOD>
  <SHORT-NAME>linearAndTexttable</SHORT-NAME>
  <CATEGORY>SCALE_LINEAR_AND_TEXTTABLE</CATEGORY>
  <UNIT-REF DEST="UNIT">kmph</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-SCALES>
  </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
    
```

```

        <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>
    
```

### 5.5.1.7 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 5.16: 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-SCALE>
            <COMPU-RATIONAL-COEFFS>
                <COMPU-NUMERATOR>
                    <V>1000</V>
                </COMPU-NUMERATOR>
                <COMPU-DENOMINATOR>
                    <V>60</V>
            </COMPU-RATIONAL-COEFFS>
        </COMPU-SCALES>
    </COMPU-PHYS-TO-INTERNAL>
</COMPU-METHOD>
    
```

```

    <v>2</v>
  </COMPU-DENOMINATOR>
</COMPU-RATIONAL-COEFFS>
</COMPU-SCALE>
</COMPU-SCALES>
</COMPU-PHYS-TO-INTERNAL>
</COMPU-METHOD>

```

### 5.5.1.8 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(00) = flat tire 01(16) = low pressure 10(32) = unbalanced 11(48) = unknown
All Bits	error	11111111 = invalid value

**Table 5.84: 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 5.17: 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>
<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>
  <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>
  <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>

```

```

<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>
  <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>

```

<b>Class</b>	<b>CompuConstTextContent</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the textual content of a scale.			
<b>Base</b>	ARObject, CompuConstContent			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
vt	VerbatimString	0..1	attr	This represents a textual constant in the computation method.

**Table 5.85: CompuConstTextContent**

<b>Class</b>	<b>CompuConstNumericContent</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the fact that the constant value of the computation method is a numerical value. It is separated from CompuConstFormulaContent to support compatibility with ASAM HDO.			
<b>Base</b>	ARObject, CompuConstContent			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
v	Numerical	0..1	attr	This represents the numerical value. <b>Tags:</b> xml.sequenceOffset=50

**Table 5.86: 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.]()



**[TPS\_SWCT\_01056] Physical dimension** [The substructure of physical dimensions defines all used quantities in the SI-System<sup>18</sup> (e.g. velocity as length/time corresponds to m/s).] ([RS\\_SWCT\\_02100](#))

**[TPS\_SWCT\_01057] Unit references one physical dimension** [The unit references one physical dimension. If the physical dimensions of two units are identical, a conversion between them is basically possible.] ([RS\\_SWCT\\_02100](#))

<b>Class</b>	<b>Unit</b>			
<b>Package</b>	M2::MSR::AsamHdo::Units			
<b>Note</b>	<p>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.</p> <p>For the calculation from SI-unit to the defined unit the factor (factorSiToUnit ) and the offset (offsetSiToUnit ) are applied as follows:</p> $x \{unit\} := y * \{siUnit\} * factorSiToUnit \{unit\}/\{siUnit\} + offsetSiToUnit \{unit\}$ <p>For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit ) and the negation of the offset (offsetSiToUnit ) are applied.</p> $y \{siUnit\} := (x*\{unit\} - offsetSiToUnit \{unit\}) / (factorSiToUnit \{unit\}/\{siUnit\})$ <p><b>Tags:</b>atp.recommendedPackage=Units</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
displayName	SingleLanguageUnit Names	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. <b>Tags:</b> xml.sequenceOffset=20
factorSiToUnit	Float	0..1	attr	This is the factor for the conversion from SI Units to units. The inverse is used for conversion from units to SI Units. <b>Tags:</b> xml.sequenceOffset=30
offsetSiToUnit	Float	0..1	attr	This is the offset for the conversion from and to siUnits. <b>Tags:</b> xml.sequenceOffset=40
physical Dimension	<a href="#">PhysicalDimension</a>	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. <b>Tags:</b> xml.sequenceOffset=50

**Table 5.87: Unit**

**[TPS\_SWCT\_01058] UnitGroup** [The [UnitGroups](#) determine if such a conversion is appropriate.] ([RS\\_SWCT\\_02100](#))

Figure 5.53 depicts the concept how units are defined.

For a detailed description of these elements please refer to the [27]. Standard units are already predefined for AUTOSAR in form of a description file.

**[TPS\_SWCT\_01736] Default values for [Unit.physicalDimension](#)** [If a [Unit](#) does not define the attribute [Unit.physicalDimension](#), the default [PhysicalDimension](#) with the [shortName](#) NoDimension applies where all physical exponents are set to 0.] ([RS\\_SWCT\\_02100](#))

<sup>18</sup>For the definition of what SI units are, see <http://physics.nist.gov/cuu/Units/>

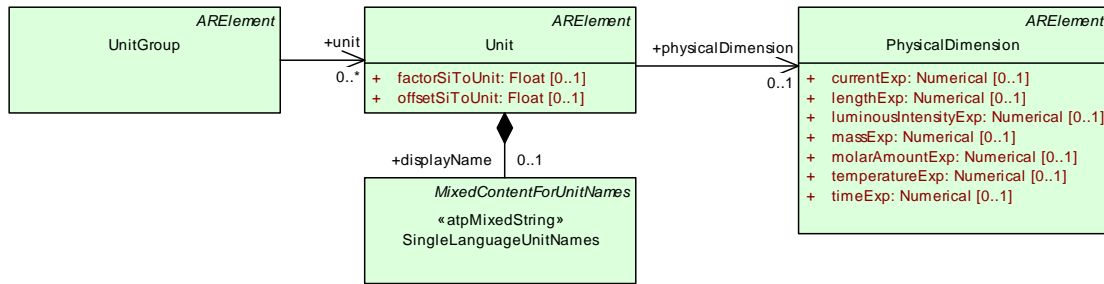


Figure 5.53: Definition of SI based units

**[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\_01737] Default values for physical exponents** [The default value of attributes `currentExp`, `lengthExp`, `luminousIntensityExp`, `massExp`, `molarAmountExp`, `temperatureExp`, `timeExp` is 0.] ([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.

<b>Class</b>	<b>PhysicalDimension</b>			
<b>Package</b>	M2::MSR::AsamHdo::Units			
<b>Note</b>	<p>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.</p> <p>Note that the equivalence of the exponents does not per se define the convertibility. For example Energy and Torque share the same exponents (Nm).</p> <p>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.</p> <p><b>Tags:</b>atp.recommendedPackage=PhysicalDimensions</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
currentExp	Numerical	0..1	attr	This attribute represents the exponent of the physical dimension "electric current". <b>Tags:</b> xml.sequenceOffset=50
lengthExp	Numerical	0..1	attr	The exponent of the physical dimension "length". <b>Tags:</b> xml.sequenceOffset=20
luminous IntensityExp	Numerical	0..1	attr	The exponent of the physical dimension "luminous intensity". <b>Tags:</b> xml.sequenceOffset=80
massExp	Numerical	0..1	attr	The exponent of the physical dimension "mass". <b>Tags:</b> xml.sequenceOffset=30

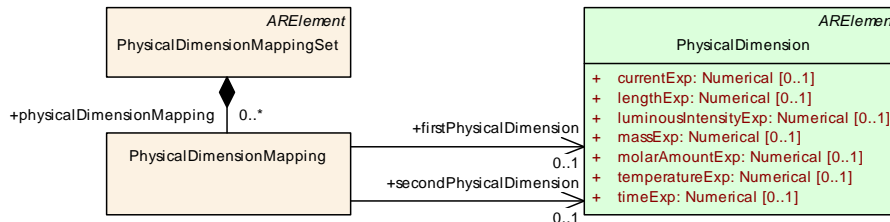




Class	PhysicalDimension			
molarAmountExp	Numerical	0..1	attr	The exponent of the physical dimension "quantity of substance". <b>Tags:</b> xml.sequenceOffset=70
temperatureExp	Numerical	0..1	attr	The exponent of the physical dimension "temperature". <b>Tags:</b> xml.sequenceOffset=60
timeExp	Numerical	0..1	attr	The exponent of the physical dimension "time". <b>Tags:</b> xml.sequenceOffset=40

**Table 5.88: PhysicalDimension**

AUTOSAR provides the ability to map two [PhysicalDimensions](#) onto each other with the implication that the two mapped [PhysicalDimensions](#) shall be considered compatible (for more explanation please refer to [\[constr\\_1053\]](#)).



**Figure 5.54: Modeling of PhysicalDimensionMapping**

[PhysicalDimensionMappings](#) are aggregated in form of [PhysicalDimensionMappingSets](#). This allows for gathering semantically related [PhysicalDimensionMappings](#) into the same [PhysicalDimensionMappingSet](#).

Class	PhysicalDimensionMappingSet			
Package	M2::MSR::AsamHdo::Units			
Note	This class represents a container for a list of mappings between PhysicalDimensions. <b>Tags:</b> atp.recommendedPackage=PhysicalDimensionMappingSets			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
physical Dimension Mapping	<a href="#">PhysicalDimensionMapping</a>	*	aggr	This aggregation represents a concrete collections of PhysicalDimensionMappings in the context of one PhysicalDimensionMappingSet.

**Table 5.89: PhysicalDimensionMappingSet**

Class	PhysicalDimensionMapping			
Package	M2::MSR::AsamHdo::Units			
Note	This class represents a specific mapping between two PhysicalDimensions.			
Base	<a href="#">ARObject</a>			
Attribute	Type	Mult.	Kind	Note





Class	PhysicalDimensionMapping			
firstPhysicalDimension	<a href="#">PhysicalDimension</a>	0..1	ref	This represents the first PhysicalDimension of the enclosing PhysicalDimensionMapping.
secondPhysicalDimension	<a href="#">PhysicalDimension</a>	0..1	ref	This represents the first PhysicalDimension of the enclosing PhysicalDimensionMapping.

**Table 5.90: PhysicalDimensionMapping**

**[constr\_1915] Existence of attribute [PhysicalDimensionMapping.firstPhysicalDimension](#)** [For each [PhysicalDimensionMapping](#), attribute [firstPhysicalDimension](#) shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1916] Existence of attribute [PhysicalDimensionMapping.secondPhysicalDimension](#)** [For each [PhysicalDimensionMapping](#), attribute [secondPhysicalDimension](#) shall exist **at the time when the contract phase generation is executed.**]()

In the following example, the units “km” and “m” and their physical dimension named “Len1” are specified. The SI base unit is “m” (Meter).

The default value of attribute [Unit.offsetSiToUnit](#) is 0, the default of [Unit.factorSiToUnit](#) is 1 (see [TPS\_SWCT\_01492]).

Given the equality 1 km == 1000 m, the following equation applies:

$$x[km] := y * [m] * 0.001[km/m] + 0[km]$$

This correlation is reflected in the example ARXML contained in Listing 5.18.

**Listing 5.18: Example for [Unit](#) and [PhysicalDimension](#)**

```

<UNIT>
  <SHORT-NAME>KiloMtr</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Kilo Meter</L-4>
  </LONG-NAME>
  <DISPLAY-NAME>km</DISPLAY-NAME>
  <FACTOR-SI-TO-UNIT>0.001</FACTOR-SI-TO-UNIT>
  <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-UNIT>
  <PHYSICAL-DIMENSION-REF BASE="PhysicalDimensions" DEST="PHYSICAL-
    DIMENSION">/SiUnit/Len1</PHYSICAL-DIMENSION-REF>
</UNIT>

<UNIT>
  <SHORT-NAME>Mtr</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Meter</L-4>
  </LONG-NAME>
  <DISPLAY-NAME>m</DISPLAY-NAME>
  <FACTOR-SI-TO-UNIT>1</FACTOR-SI-TO-UNIT>
  <OFFSET-SI-TO-UNIT>0</OFFSET-SI-TO-UNIT>

```

```

<PHYSICAL-DIMENSION-REF BASE="PhysicalDimensions" DEST="PHYSICAL-
  DIMENSION"/>/SiUnit/Len1</PHYSICAL-DIMENSION-REF>
</UNIT>

<PHYSICAL-DIMENSION>
  <SHORT-NAME>Len1</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Length 1</L-4>
  </LONG-NAME>
  <LENGTH-EXP>1</LENGTH-EXP>
</PHYSICAL-DIMENSION>

```

**[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`.

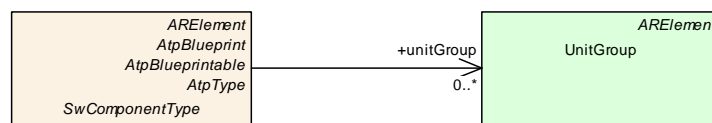
This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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 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 General Blueprints [28] 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 **at any time in the workflow.**]()

**[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 choose the most appropriate one.] (*RS\_SWCT\_02100*)



**Figure 5.55: Relation of SwComponentType to UnitGroup**

<b>Class</b>	<b>UnitGroup</b>			
<b>Package</b>	M2::MSR::AsamHdo::Units			
<b>Note</b>	<p>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.</p> <p>In this way, e.g. country-specific unit systems (CATEGORY="COUNTRY") can be defined as well as specific unit systems for certain application domains.</p> <p>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".</p> <p>Note that the UnitGroup does not ensure the physical compliance of the units. This is maintained by the physical dimension.</p> <p><b>Tags:</b>atp.recommendedPackage=UnitGroups</p>			
<b>Base</b>	<i>ARElement, ARObjct, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
unit	Unit	*	ref	<p>This represents one particular unit in the UnitGroup.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>

**Table 5.91: 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 [27] the following three values for `category`s 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 5.56, `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 "KmPerHour" 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" in this specific example.

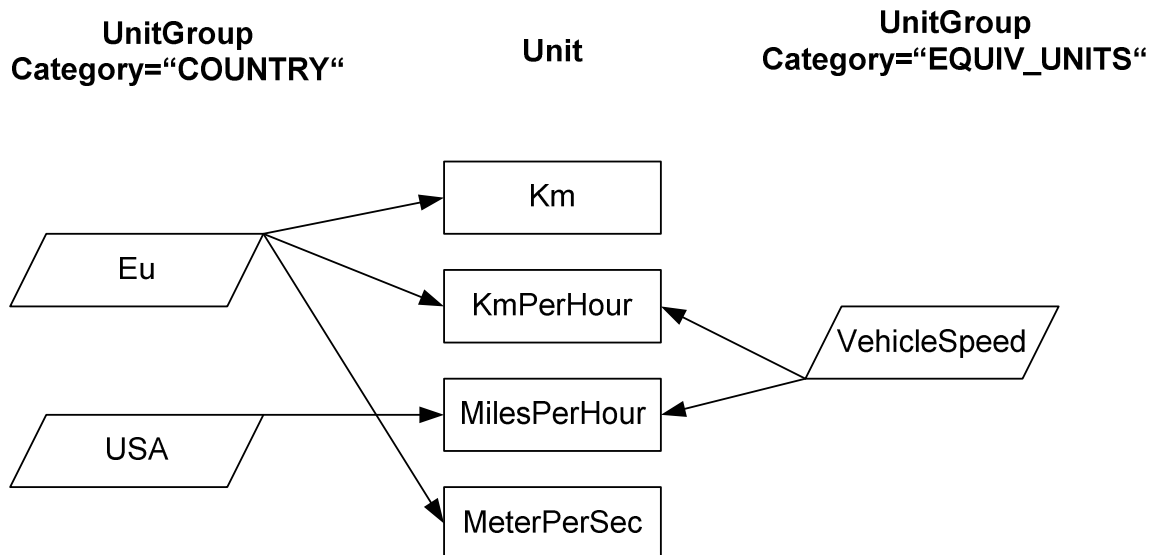


Figure 5.56: Example for units and unit groups

### 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 5.57 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 [27]. 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.

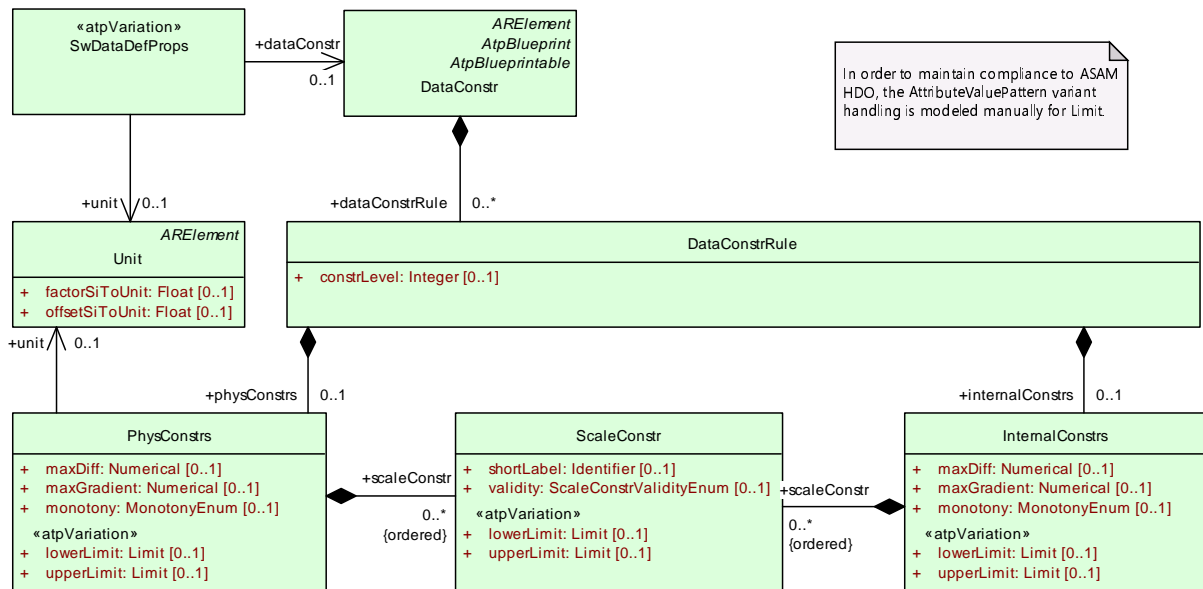
This rule applies **at any time in the workflow**. Other values may exist, but the semantics is outside the AUTOSAR scope.

]()

**[TPS\_SWCT\_01287] Standard limits and extended limits in the ASAM-MCD2 (ASAP2) specification** [The ASAM-MCD2 (ASAP2) specification [26] defines 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 counter-intuitive (since the one with `constrLevel` set to 0 changes its role), it matches the best to the definitions in ASAM-MCD2.]()



**Figure 5.57: Meta-model for defining Data Constraints**

<b>Class</b>	<b>DataConstr</b>			
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to specify constraints on data. <b>Tags:</b> atp.recommendedPackage=DataConstrs			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> , <a href="#">Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>







Class	DataConstr			
dataConstrRule	<a href="#">DataConstrRule</a>	*	aggr	<p>This is one particular rule within the data constraints.</p> <p><b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false</p>

**Table 5.92: DataConstr**

Class	DataConstrRule			
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to express one specific data constraint rule.			
<b>Base</b>	ARObject			
Attribute	Type	Mult.	Kind	Note
constrLevel	Integer	0..1	attr	<p>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.</p> <p>The lower the level, the more stringent the check.</p> <p>Used to distinguish hard or soft limits.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>
internalConstrs	<a href="#">InternalConstrs</a>	0..1	aggr	<p>Describes the limitations applicable on the internal domain (as opposed to the physical domain).</p> <p><b>Tags:</b>xml.sequenceOffset=40</p>
physConstrs	<a href="#">PhysConstrs</a>	0..1	aggr	<p>Describes the limitations applicable on the physical domain (as opposed to the internal domain).</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>

**Table 5.93: DataConstrRule**

Class	PhysConstrs			
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to express physical constraints. Therefore it has (in opposite to InternalConstrs) a reference to a Unit.			
<b>Base</b>	ARObject			
Attribute	Type	Mult.	Kind	Note
lowerLimit	<a href="#">Limit</a>	0..1	attr	<p>This specifies the lower limit of the constraint.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=20</p>
maxDiff	Numerical	0..1	attr	<p>Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis.</p> <p><b>Tags:</b>xml.sequenceOffset=60</p>
maxGradient	Numerical	0..1	attr	<p>This element specifies the maximum slope that may be used in curves and maps.</p> <p><b>Tags:</b>xml.sequenceOffset=50</p>





Class	PhysConstrs			
monotony	<a href="#">MonotonyEnum</a>	0..1	attr	This specifies the monotony constraints on the data object. Note that this applies only to curves and maps. <b>Tags:</b> xml.sequenceOffset=70
scaleConstr (ordered)	<a href="#">ScaleConstr</a>	*	aggr	This is one particular scale which contributes to the data constraints. <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false
unit	<a href="#">Unit</a>	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. <b>Tags:</b> xml.sequenceOffset=80
upperLimit	<a href="#">Limit</a>	0..1	attr	This specifies the upper limit of the constraint. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

**Table 5.94: PhysConstrs**

Class	InternalConstrs			
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to express internal constraints.			
<b>Base</b>	<i>ARObject</i>			
Attribute	Type	Mult.	Kind	Note
lowerLimit	<a href="#">Limit</a>	0..1	attr	This specifies the lower limit of the constraint. <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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. <b>Tags:</b> xml.sequenceOffset=60
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in maps and curves. <b>Tags:</b> xml.sequenceOffset=50
monotony	<a href="#">MonotonyEnum</a>	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, "no Monotony" is the default content. <b>Tags:</b> xml.sequenceOffset=70





Class	InternalConstrs			
scaleConstr (ordered)	<a href="#">ScaleConstr</a>	*	aggr	This is one particular scale which contributes to the data constraints.  <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false
upperLimit	<a href="#">Limit</a>	0..1	attr	This specifies the upper limit defined by the constraint.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

**Table 5.95: InternalConstrs**

Class	ScaleConstr			
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to specify constraints as a list of intervals (called scales).			
<b>Base</b>	ARObject			
Attribute	Type	Mult.	Kind	Note
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question.  <b>Tags:</b> xml.sequenceOffset=30
lowerLimit	<a href="#">Limit</a>	0..1	attr	This specifies the lower limit of the scale.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
shortLabel	Identifier	0..1	attr	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.  <b>Tags:</b> xml.sequenceOffset=20
upperLimit	<a href="#">Limit</a>	0..1	attr	This specifies the upper limit of a the scale.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
validity	<a href="#">ScaleConstrValidity Enum</a>	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".  <b>Tags:</b> xml.attribute=true

**Table 5.96: ScaleConstr**

Enumeration	ScaleConstrValidityEnum
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints
<b>Note</b>	This enumerator specifies the possible values of a scale.
<b>Literal</b>	<b>Description</b>





Enumeration	ScaleConstrValidityEnum
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. <b>Tags:</b> atp.EnumerationLiteralIndex=0
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. <b>Tags:</b> atp.EnumerationLiteralIndex=1
notValid	The ECU cannot process the requested data. <b>Tags:</b> atp.EnumerationLiteralIndex=2
valid	Current value is within a valid range and can be presented to user as is. <b>Tags:</b> atp.EnumerationLiteralIndex=3

**Table 5.97: ScaleConstrValidityEnum**

Primitive	Limit			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	This class represents the ability to express a numerical limit. Note that this is in fact a NumericalVariation Point but has the additional attribute intervalType. <b>Tags:</b> xml.xsd.customType=LIMIT-VALUE xml.xsd.pattern=(0[xX][0-9a-fA-F+]) (0[0-7]+) (0[bB][0-1]+) (([\+\-]?[1-9][0-9]+\.[0-9]+)? [\+\-]?[0-9][\.[0-9]+)? ([eE]([\+\-]?[0-9]+)? \.0 INF -INF NaN) xml.xsd.type=string			
Attribute	Type	Mult.	Kind	Note
intervalType	<a href="#">IntervalTypeEnum</a>	0..1	attr	This specifies the type of the interval. If the attribute is missing the interval shall be considered as "CLOSED". <b>Tags:</b> xml.attribute=true

**Table 5.98: Limit**

Enumeration	MonotonyEnum
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
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. <b>Tags:</b> atp.EnumerationLiteralIndex=0
increasing	This indicates that the related curve needs to be monotony increasing. <b>Tags:</b> atp.EnumerationLiteralIndex=1
monotonous	This indicates that the values shall be monotonously decreasing or increasing, depending on the trend set by the first values of the series. <b>Tags:</b> atp.EnumerationLiteralIndex=2
noMonotony	This indicates that the related curve needs not to be monotony. <b>Tags:</b> atp.EnumerationLiteralIndex=3
strictlyDecreasing	This indicates that the related curve needs to be strictly monotony decreasing. <b>Tags:</b> atp.EnumerationLiteralIndex=4
strictlyIncreasing	This indicates that the related curve needs to be strictly monotony increasing. <b>Tags:</b> atp.EnumerationLiteralIndex=5





Enumeration	MonotonyEnum
strictMonotonous	This indicates that the values shall be strict monotonously decreasing or increasing, depending on the trend set by the first values of the series. <b>Tags:</b> atp.EnumerationLiteralIndex=6

**Table 5.99: 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 ApplicationDataType 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 only PhysConstrs are provided to ApplicationDataTypes the CompuMethod can be used to compute the InternalConstrs.]()

**[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:

$$\begin{aligned}
 range = & \{x \in \mathbb{R} \mid lowerLimit.value < x < upperLimit.value\} \\
 & \cup \{lowerLimit.value \mid lowerLimit.intervalType == "CLOSED"\} \\
 & \cup \{upperLimit.value \mid upperLimit.intervalType == "CLOSED"\}
 \end{aligned}$$

]()

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 **at any time in the workflow.**]()

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 the value provided as a limit actually makes sense in any of the given contexts.

<i>Enumeration</i>	<b>IntervalTypeEnum</b>
<i>Package</i>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
<i>Note</i>	This enumerator specifies the type of an interval.
<i>Literal</i>	<i>Description</i>
closed	The area is limited by the value given. The value itself is included. <b>Tags:</b> atp.EnumerationLiteralIndex=0
open	The area is limited by the value given. The value itself is not included. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 5.100: IntervalTypeEnum**

### 5.5.3.1 Physical Limits

Physical limits can be given at various places in the AUTOSAR Meta-Model, e.g. in context of [ApplicationDataTypes](#), [DataPrototypes](#) but also without the usage of the type prototype pattern in [Compound Primitive Data Types](#) (e.g. [SwAxisIndividual.dataConstr](#)).

Nevertheless, the usage of [PhysConstrs](#) requires a [CompuMethod](#) for the calculation of the numerical limits, which cannot be applied for textual conversions. For this reason following definition applies:

**[TPS\_SWCT\_01761] Physical limits of pure textual conversions** [It is not possible to define the lower or upper limit of a set of textual labels. Therefore, it is not possible to define limits for an object that can only take elements of a set of textual labels as the value.] ()

Please note, as a consequence of [\[TPS\\_SWCT\\_01761\]](#) for data defined by means of a [CompuMethod](#) of [category TEXTTABLE](#) or [BITFIELD\\_TEXTTABLE](#) and additionally a [DataConstr](#) with a [dataConstrRule.physConstrs](#) the given [physConstrs](#) has no meaning.

**[TPS\_SWCT\_01762] Physical limits of mixed textual conversions** [The definition of the physical limits of a piece of data described by a [CompuMethod](#) of [category SCALE\\_LINEAR\\_AND\\_TEXTTABLE](#) and [SCALE\\_RATIONAL\\_AND\\_TEXTTABLE](#) can only be specified for the **linear** or **rational** part.

In addition, the defined textual labels can be used for the conversion.] ()

For clarification, [\[TPS\\_SWCT\\_01761\]](#) and [\[TPS\\_SWCT\\_01762\]](#) do not limit the usage of [DataConstr.dataConstrRule.internalConstrs](#) which may define further and even tighter constraints on implementation level.

Those [internalConstrs](#) might be even given in context of a [Compound Primitive Data Type](#) (for example, in the context of an [SwAxisIndividual.inputVariableType](#) or [SwAxisIndividual.dataConstr](#)).

### 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.]()

<b>Class</b>	<b>MemorySection</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::MemorySectionUsage
<b>Note</b>	<p>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.</p> <p>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:</p> <pre>&lt;SwAddrMethod shortName&gt;[_&lt;further specialization nominator&gt;][_&lt;alignment&gt;]</pre> <p>where</p> <ul style="list-style-type: none"> <li>• [<b>&lt;SwAddrMethod shortName&gt;</b>] is the shortName of the referenced SwAddrMethod</li> <li>• [<b>&lt;further specialization nominator&gt;</b>] 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.</li> <li>• [<b>&lt;alignment&gt;</b>] is the alignment attributes value and is only applicable in the case that the memoryAllocationKeywordPolicy value of the referenced SwAddrMethod is set to addrMethod ShortNameAndAlignment</li> </ul> <p>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.</p> <p>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 BswModule Description resp. the SwComponentType. It can be superseded by the prefix attribute.</p>
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>





Class		MemorySection		
Attribute	Type	Mult.	Kind	Note
alignment	<a href="#">AlignmentType</a>	0..1	attr	The attribute describes the typical alignment of objects within this memory section.
executableEntity	<a href="#">ExecutableEntity</a>	*	ref	Reference to the ExecutableEntities located in this section. This allows to locate different Executable Entities in different sections even if the associated Sw Addrmethod is the same.  This is applicable to code sections only.
memClass Symbol	CIdentifier	0..1	attr	Defines a specific symbol in order to generate the compiler abstraction "memclass" code for this Memory Section. 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.  <b>Tags:</b> atp.Status=obsolete
option	Identifier	*	attr	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): <ul style="list-style-type: none"> <li>• INLINE - The code section is declared with the compiler abstraction macro INLINE.</li> <li>• LOCAL_INLINE - The code section is declared with the compiler abstraction macro LOCAL_INLINE</li> </ul> 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 Bsw ModuleDescription'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.
swAddrmethod	<a href="#">SwAddrMethod</a>	0..1	ref	This association indicates that this module specific (abstract) memory section is part of an overall SwAddr Method, 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.







Class	MemorySection			
symbol	Identifier	0..1	attr	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 sectionName Prefixes.

**Table 5.101: MemorySection**

**[constr\_10033] Existence of `MemorySection.swAddrmethod`** [For each `MemorySection`, attribute `swAddrmethod` shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_10034] Existence of `MemorySection.alignment`** [For each `MemorySection`, attribute `alignment` shall exist **at the time when the contract phase generation is executed** if the attribute `MemorySection.swAddrmethod.memoryAllocationKeywordPolicy` is set to `MemoryAllocationKeywordPolicyType.addrMethodShortNameAndAlignment`.]()

Rationale for the existence of [constr\_10033]: If the `MemorySection` is configured such that `MemorySection.swAddrmethod.memoryAllocationKeywordPolicy` is set to `MemoryAllocationKeywordPolicyType.addrMethodShortNameAndAlignment`, then the `alignment` attribute is mandatory because the implementation of the memory mapping shall be able to formally consider the alignment for the choice of memory sections.

Class	SwAddrMethod			
Package	M2::MSR::DataDictionary::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. <b>Tags:</b> atp.recommendedPackage=SwAddrMethods			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
memory Allocation KeywordPolicy	<a href="#">MemoryAllocationKeywordPolicyType</a>	0..1	attr	Enumeration to specify the name pattern of the Memory Allocation Keyword.
option	Identifier	*	attr	This attribute introduces the ability to specify further intended properties of the <code>MemorySection</code> 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 <code>MemMapAddressingModeSet</code> .





Class	SwAddrMethod			
section Initialization Policy	<a href="#">SectionInitialization PolicyType</a>	0..1	attr	Specifies the expected initialization of the variables (inclusive those which are implementing VariableData Prototypes). 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 AutosarData Prototypes 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	<a href="#">MemorySectionType</a>	0..1	attr	Defines the type of memory sections which can be associated with this addressing method.

**Table 5.102: SwAddrMethod**

**[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.

**configClassPreBuild** This corresponds to config data which is assigned at pre-compile or link time.

**configClassPostBuild** This corresponds to config data which is assigned at post-build time.

]()

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 [29] 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`** [Any given collection of values stored in the attributes `MemorySection.option` or `SwAddrMethod.option` according to [TPS\_SWCT\_01456] shall at most include a single value out of the following list **at the time when the RTE is generated:**

- `safetyQM`
- `safetyAsilA`
- `safetyAsilB`
- `safetyAsilC`
- `safetyAsilD`

]()

**[constr\_1381] Appearance of core-related possible values of `MemorySection.option` or `SwAddrMethod.option`** [Any given collection of values stored in the attributes `MemorySection.option` or `SwAddrMethod.option` according to [TPS\_SWCT\_01456] shall at most include a single value out of the following list **at the time when the RTE is generated:**

- `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.]()

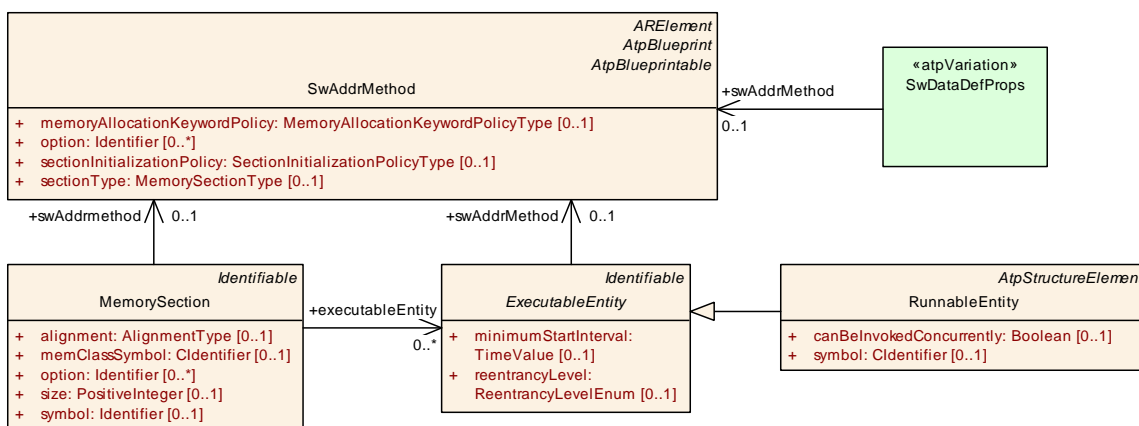
Figure 5.59 illustrates the usage of SwAddrMethod in the context of a DataPrototype.

**[TPS\_SWCT\_01292] Usage of SwAddrMethod in the context of a DataPrototype** [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 RunnableEntities and BswSchedulableEntities from the shortName of the SwAddrMethod only because the alignment defined in MemorySection is not known at contract phase.]()

**[constr\_2034] SwAddrMethod referenced by RunnableEntities, BswCalledEntities, or BswSchedulableEntities** [RunnableEntities, BswCalledEntities, and BswSchedulableEntities shall not reference a SwAddrMethod which attribute memoryAllocationKeywordPolicy is set to addrMethodShortNameAndAlignment at the time when the RTE is generated.]()



**Figure 5.58: SwAddrMethod**

**[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` **at the time when the RTE is generated.**]()

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.

<b>Primitive</b>	<b>SectionInitializationPolicyType</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
<b>Note</b>	<p>SectionInitializationPolicyType describes the intended initialization of MemorySections. The following values are standardized in AUTOSAR Methodology:</p> <ul style="list-style-type: none"> <li>• <b>INIT</b>: To be used for (explicitly or not explicitly) initialized variables.</li> <li>• <b>CLEARED</b>: To be used for not explicitly initialized variables.</li> <li>• <b>POWER-ON-CLEARED</b>: To be used for variables that are not explicitly initialized (cleared) during normal start-up. Instead these are cleared only after power on reset.</li> </ul> <p>Please note that the values are defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility.</p> <p><b>Tags:</b> xml.xsd.customType=SECTION-INITIALIZATION-POLICY-TYPE xml.xsd.type=NMTOKEN</p>

**Table 5.103: SectionInitializationPolicyType**

**[constr\_10068] Standardized values for `SectionInitializationPolicyType`** [The following values for `SectionInitializationPolicyType` are reserved by the AUTOSAR standard:

**INIT** To be used for (explicitly or not explicitly) initialized variables.

**CLEARED** To be used for (explicitly or not explicitly) initialized variables.

**POWER-ON-CLEARED** To be used for variables that are not explicitly initialized (cleared) during normal start-up. Instead these are cleared only after power on reset.

This rule shall be imposed **at the time when the RTE is generated.**]()

Please note that custom values of `SectionInitializationPolicyType` are currently not supported.

<b>Enumeration</b>	<b>MemorySectionType</b>
<b>Package</b>	M2::MSR::DataDictionary::AuxillaryObjects
<b>Note</b>	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.
<b>Literal</b>	<b>Description</b>





<b>Enumeration</b>	<b>MemorySectionType</b>
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. <b>Tags:</b> atp.EnumerationLiteralIndex=2
calprm	To be used for calibratable constants of ECU-functions. <b>Tags:</b> atp.EnumerationLiteralIndex=3
code	To be used for mapping code to application block, boot block, external flash etc. <b>Tags:</b> atp.EnumerationLiteralIndex=4
configData	Constants with attributes that show that they reside in one segment for module configuration. <b>Tags:</b> atp.EnumerationLiteralIndex=5
const	To be used for global or static constants. <b>Tags:</b> atp.EnumerationLiteralIndex=6
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. <b>Tags:</b> atp.EnumerationLiteralIndex=7
var	To be used for global or static variables. The expected initialization is specified with the attribute sectionInitializationPolicy. <b>Tags:</b> atp.EnumerationLiteralIndex=9

**Table 5.104: MemorySectionType**

<b>Enumeration</b>	<b>MemoryAllocationKeywordPolicyType</b>
<b>Package</b>	M2::MSR::DataDictionary::AuxillaryObjects
<b>Note</b>	Enumeration to specify the name pattern of the Memory Allocation Keyword.
<b>Literal</b>	<b>Description</b>
addrMethodShort Name	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. <b>Tags:</b> atp.EnumerationLiteralIndex=0
addrMethodShort NameAndAlignment	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. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 5.105: MemoryAllocationKeywordPolicyType**

<b>Primitive</b>	<b>AlignmentType</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes

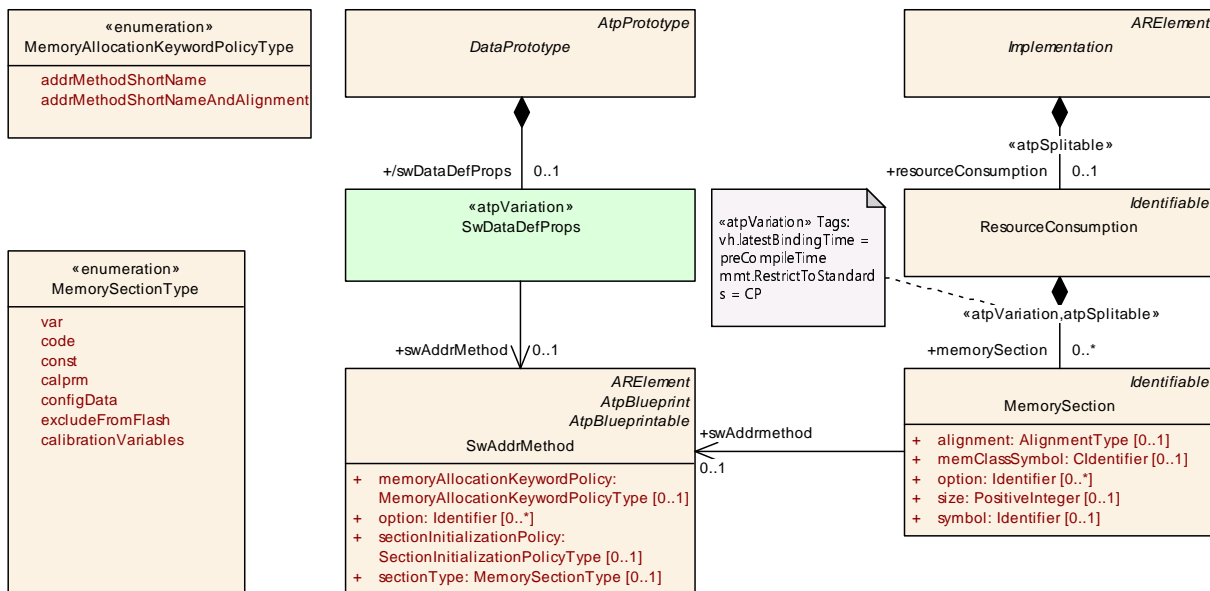




Primitive	AlignmentType
<b>Note</b>	This primitive represents the alignment of objects within a memory section. The value is in number of bits or UNKNOWN (deprecated), 8, 16, 32 UNSPECIFIED, BOOLEAN, or PTR. Typical values for numbers are 8, 16, 32.  <b>Tags:</b> xml.xsd.customType=ALIGNMENT-TYPE xml.xsd.pattern=[1-9][0-9]* 0[xX][0-9a-fA-F]* 0[bB][0-1]+ 0[0-7]* UNSPECIFIED UNKNOWN BOOLEAN PTR xml.xsd.type=string

**Table 5.106: AlignmentType**

For more information on the specification of a [MemorySection](#) refer to [6].



**Figure 5.59: 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

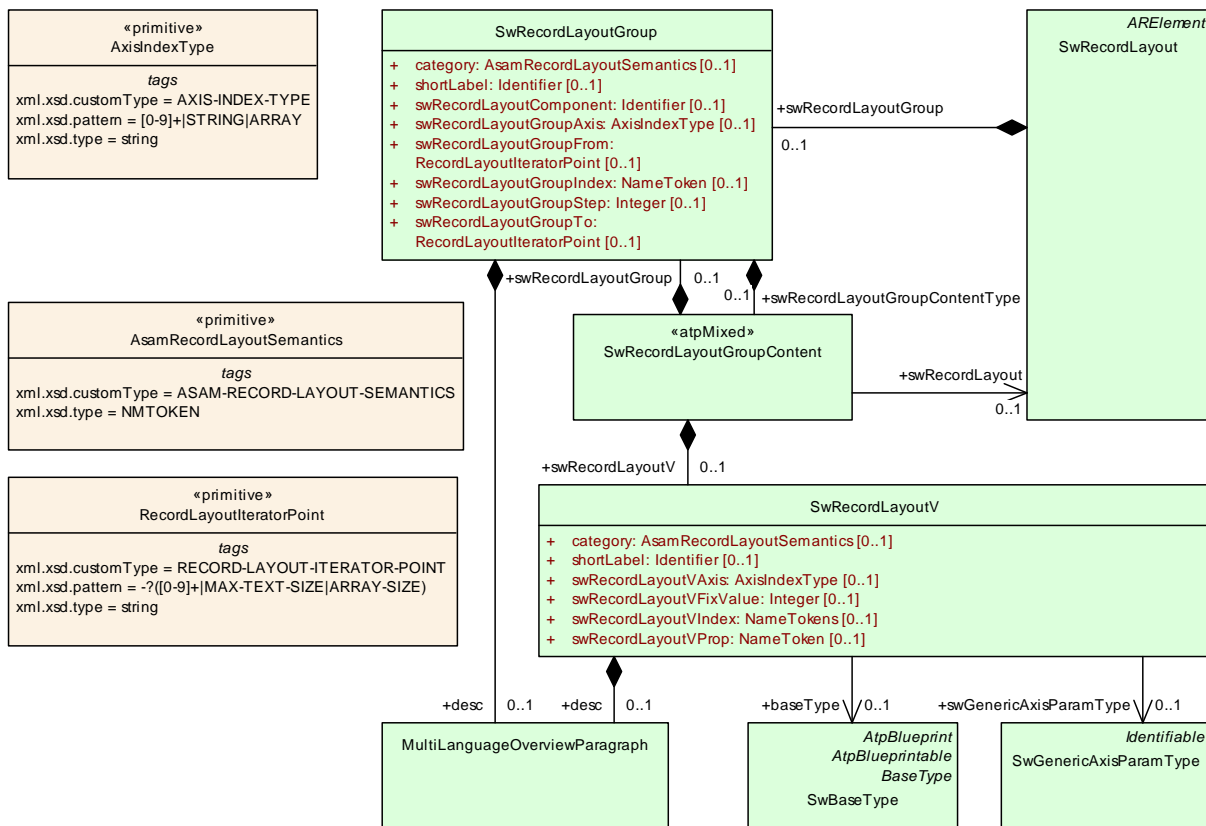
these may refer to the same `SwRecordLayout` even if the size of the data is different.] ([RS\\_SWCT\\_03215](#))

### 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 5.60: Specification of a record layout**



<b>Class</b>	<b>SwRecordLayout</b>			
<b>Package</b>	M2::MSR::DataDictionary::RecordLayout			
<b>Note</b>	<p>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.</p> <p><b>Tags:</b>atp.recommendedPackage=SwRecordLayouts</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swRecordLayoutGroup	<a href="#">SwRecordLayoutGroup</a>	0..1	aggr	<p>This is the top level record layout group.</p> <p><b>Tags:</b>  xml.roleElement=true  xml.roleWrapperElement=false  xml.sequenceOffset=20  xml.typeElement=false  xml.typeWrapperElement=false</p>

**Table 5.107: SwRecordLayout**

<b>Class</b>	<b>SwRecordLayoutV</b>			
<b>Package</b>	M2::MSR::DataDictionary::RecordLayout			
<b>Note</b>	<p>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.</p>			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
baseType	<a href="#">SwBaseType</a>	0..1	ref	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>
category	<a href="#">AsamRecordLayoutSemantics</a>	0..1	attr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=5</p>
desc	<a href="#">MultiLanguageOverviewParagraph</a>	0..1	aggr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>
shortLabel	<a href="#">Identifier</a>	0..1	attr	<p>This attribute specifies a name which can be used e.g. when ECU code is generated from the record layout value.</p> <p><b>Tags:</b>xml.sequenceOffset=3</p>





Class		SwRecordLayoutV		
swGenericAxisParamType	<a href="#">SwGenericAxisParamType</a>	0..1	ref	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=70</p>
swRecordLayoutVAxis	<a href="#">AxisIndexType</a>	0..1	attr	<p>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.</p> <p>In case of nested iterators (mainly for multidimensional objects) the iterator names are specified as whitespace-separated names.</p> <p>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.</p> <p>It is considered an error if more components are specified than axes exist in the related ApplicationDataType.</p> <p><b>Tags:</b>xml.sequenceOffset=40</p>
swRecordLayoutVFixValue	Integer	0..1	attr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=80</p>
swRecordLayoutVIndex	NameTokens	0..1	attr	<p>The symbolic value for iteration, or the symbolic values separated by whitespaces, refer to the symbolic values given in swRecordLayoutGroupIndex .</p> <p>The iterators are processed from left to right, in such a manner that they symbolize the loop index from the outside to the inside.</p> <p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=60</p>
swRecordLayoutVProp	NameToken	0..1	attr	<p>This attribute describes the kind of values to be stored. More details see below. The standardized values foreseen for this attribute are defined in <a href="#">[TPS_SWCT_01489]</a>.</p> <p><b>Tags:</b>xml.sequenceOffset=50</p>

**Table 5.108: SwRecordLayoutV**

Class		SwRecordLayoutGroup		
Package	M2::MSR::DataDictionary::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	Type	Mult.	Kind	Note





Class	SwRecordLayoutGroup			
category	<a href="#">AsamRecordLayoutSemantics</a>	0..1	attr	<p>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.</p> <p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=5</p>
desc	MultiLanguageOverviewParagraph	0..1	aggr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>
shortLabel	Identifier	0..1	attr	<p>This attribute specifies a name which can be used e.g. when ECU code is generated from the record layout group.</p> <p><b>Tags:</b>xml.sequenceOffset=3</p>
swGenericAxisParamType	<a href="#">SwGenericAxisParamType</a>	0..1	ref	<p>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.</p> <p>Obviously, the axis referred to by swRecordLayoutGroup Axis shall be a generic axis in which the referenced SwGenericAxisType is aggregated.</p> <p><b>Tags:</b>xml.sequenceOffset=50</p>
swRecordLayoutComponent	Identifier	0..1	attr	<p>This attribute is used to denote the component to which the group in question applies. Thus, the record layout supports structured objects.</p> <p>This secures independence from the sequence of components, because they can be referred to via name.</p> <p><b>Tags:</b>xml.sequenceOffset=90</p>
swRecordLayoutGroupAxis	<a href="#">AxisIndexType</a>	0..1	attr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>
swRecordLayoutGroupContentType	<a href="#">SwRecordLayoutGroupContent</a>	0..1	aggr	<p>This is the contents of the recordLayout which is produced for every step of iteration.</p> <p><b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=100 xml.typeElement=false xml.typeWrapperElement=false</p>
swRecordLayoutGroupFrom	<a href="#">RecordLayoutIteratorPoint</a>	0..1	attr	<p>This attribute specifies the iterator index for the point in the axis from which a record layout group is commenced.</p> <p>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'.</p> <p><b>Tags:</b>xml.sequenceOffset=60</p>





Class	SwRecordLayoutGroup			
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. <b>Tags:</b> xml.sequenceOffset=40
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". <b>Tags:</b> xml.sequenceOffset=80
swRecordLayoutGroupTo	<a href="#">RecordLayoutIteratorPoint</a>	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. <b>Tags:</b> xml.sequenceOffset=70

**Table 5.109: SwRecordLayoutGroup**

Class	<<atpMixed>> SwRecordLayoutGroupContent			
Package	M2::MSR::DataDictionary::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	Type	Mult.	Kind	Note
swRecordLayout	<a href="#">SwRecordLayout</a>	0..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. <b>Tags:</b> xml.sequenceOffset=110
swRecordLayoutGroup	<a href="#">SwRecordLayoutGroup</a>	0..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. <b>Tags:</b> xml.sequenceOffset=130
swRecordLayoutV	<a href="#">SwRecordLayoutV</a>	0..1	aggr	Particular Value specification for this record layout group. <b>Tags:</b> xml.sequenceOffset=120

**Table 5.110: 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](#).

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

For [CURVE](#), [MAP](#), etc. the iteration shall be performed along the input axis.

<b>Primitive</b>	<b>AxisIndexType</b>
<b>Package</b>	M2::MSR::DataDictionary::RecordLayout
<b>Note</b>	<p>This meta-class specifies an axis in a curve/map data object. The index satisfies the following convention:</p> <ul style="list-style-type: none"> <li>• 0 output "axis"</li> <li>• 1 input axis 1 (X input axis e.g. of a CURVE)</li> <li>• 2 input axis 2 (Y input axis e.g. of a MAP)</li> <li>• 3 input axis 3 (Z input axis e.g. of a CUBOID)</li> <li>• 4 input axis 3 (Z4 input axis e.g. of a CUBE_4)</li> <li>• 5 input axis 3 (Z5 input axis e.g. of a CUBE_5)</li> <li>• 6..9 etc.</li> </ul> <p>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.</p> <p>In addition to this, the Values STRING and ARRAY support specific iterations.</p> <p><b>Tags:</b>  xml.xsd.customType=AXIS-INDEX-TYPE  xml.xsd.pattern=[0-9]+ STRING ARRAY  xml.xsd.type=string</p>

**Table 5.111: AxisIndexType**

<b>Primitive</b>	<b>RecordLayoutIteratorPoint</b>
<b>Package</b>	M2::MSR::DataDictionary::RecordLayout
<b>Note</b>	<p>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.</p> <p><b>Tags:</b>  xml.xsd.customType=RECORD-LAYOUT-ITERATOR-POINT  xml.xsd.pattern=-?[0-9]+ MAX-TEXT-SIZE ARRAY-SIZE  xml.xsd.type=string</p>

**Table 5.112: 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 5.113.] ([RS\\_SWCT\\_03215](#))

Property	Description
<b>VALUE</b>	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.
<b>COUNT</b>	The amount of values of the axis.
<b>LEFTDIFF</b>	The difference to the previous axis point.
<b>RIGHTDIFF</b>	The difference to the next axis point.
<b>DIST</b>	The distance value of this axis in case of a fixed axis with distance specification.

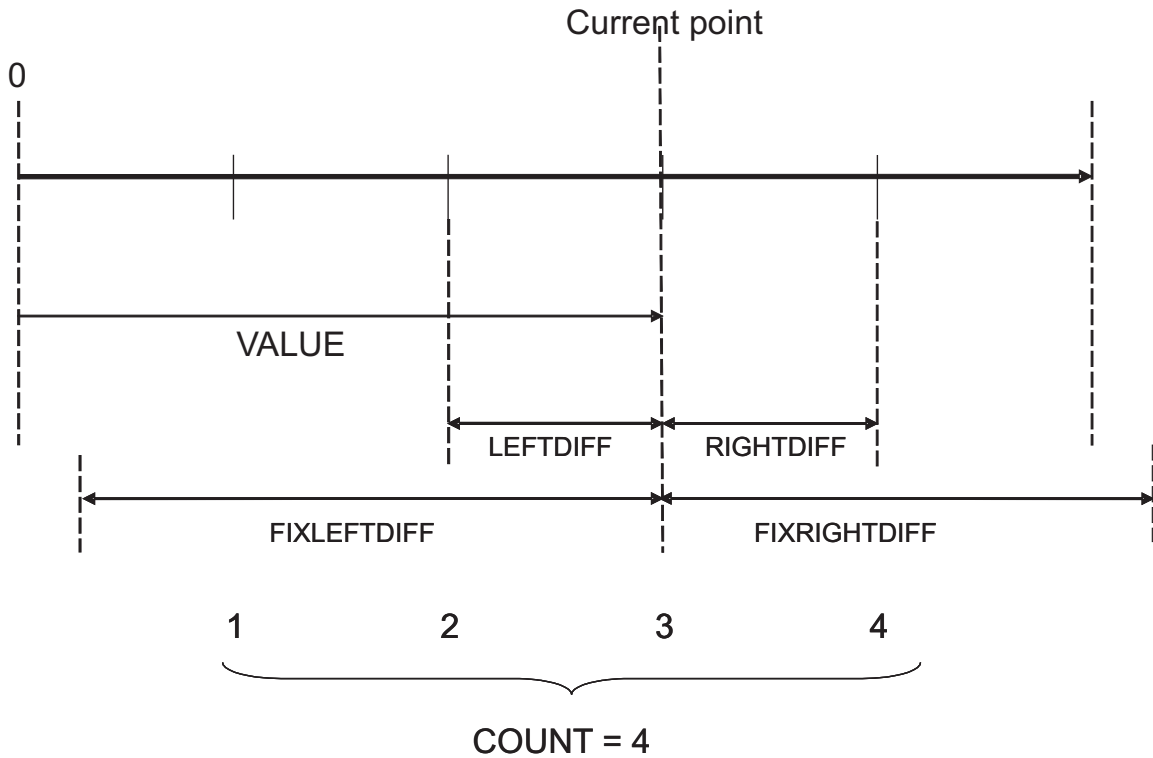




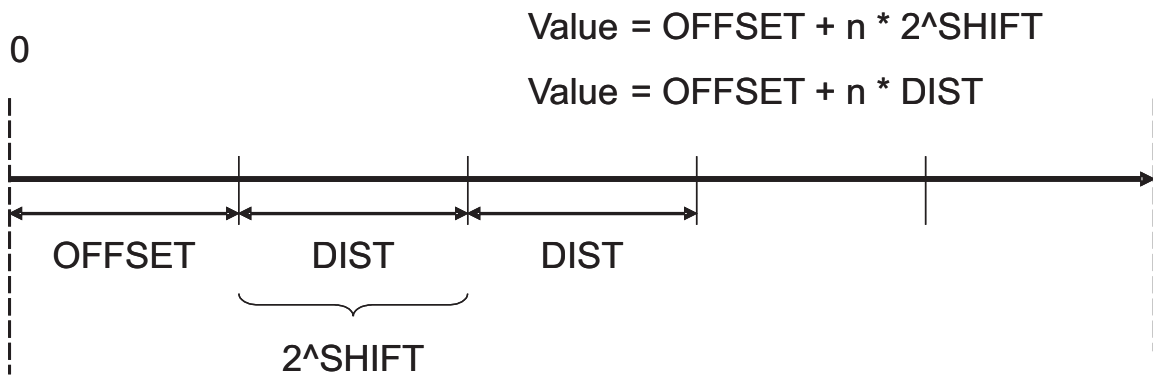
<b>SHIFT</b>	The shift value of this axis in case of a fixed axis with shift/offset.
<b>OFFSET</b>	The offset value of this axis in case of a fixed axis with shift/offset.
<b>SOURCE-ADR</b>	The address of the source of this axis (Note that this does not apply to the value axis).
<b>RESULT-ADR</b>	The address of the result for this axis (note that this does not apply to input axis).
<b>ADDRESS</b>	The address of the axis point.
<b>FILL</b>	Fill with the hex value specified as contents of <code>swRecordLayoutVFixValue</code> .
<b>FIXLEFTDIFF</b>	Difference between this and a fixed left-hand value specified in <code>swRecordLayoutVFixValue</code> .
<b>FIXRIGHTDIFF</b>	Difference between this and a fixed right-hand value specified in <code>swRecordLayoutVFixValue</code> .

**Table 5.113: swRecordLayoutVProp**

Figure 5.61 and Figure 5.62 illustrate most of these properties.



**Figure 5.61: Values for swRecordLayoutVProp for individual axis**



**Figure 5.62: 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, or 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. ] ([RS\\_SWCT\\_03215](#))

<b>Primitive</b>	<b>AsamRecordLayoutSemantics</b>
<b>Package</b>	M2::MSR::DataDictionary::RecordLayout
<b>Note</b>	<p>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.</p> <p>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.</p> <p>It is specified as NMTOKEN to reduce the direct dependency of ASAM an AUTOSAR standards.</p> <p><b>Tags:</b> xml.xsd.customType=ASAM-RECORD-LAYOUT-SEMANTICS xml.xsd.type=NMTOKEN</p>

**Table 5.114: AsamRecordLayoutSemantics**

The values of [SwRecordLayoutV.category](#) or [SwRecordLayoutGroup.category](#) can, for example, be taken from the ASAM MCD 2D specification provided in [26]. Examples are:

- INDEX\_INCR
- INDEX\_DECR
- COLUMN\_DIR
- ROW\_DIR
- ALTERNATE\_WITH\_X
- ALTERNATE\_WITH\_Y
- ALTERNATE\_CURVES

The consistency of these values of [SwRecordLayoutV.category](#) or [SwRecordLayoutGroup.category](#) with the structure of the [SwRecordLayout](#) shall be ensured by the author of the [SwRecordLayout](#).

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 5.19: 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>
  <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>
    
```

### 5.5.5.2 RecordLayouts and DataTypes

[TPS\_SWCT\_01837] Types for record layouts [Because [ParameterDataPrototypes](#) have a `<<isOfType>>`-relation to [ApplicationDataTypes](#) or [ImplementationDataTypes](#), the related data types shall properly match to the details as specified in [swDataDefProps](#).] ([RS\\_SWCT\\_03215](#))

This is exemplified in figure 5.63.



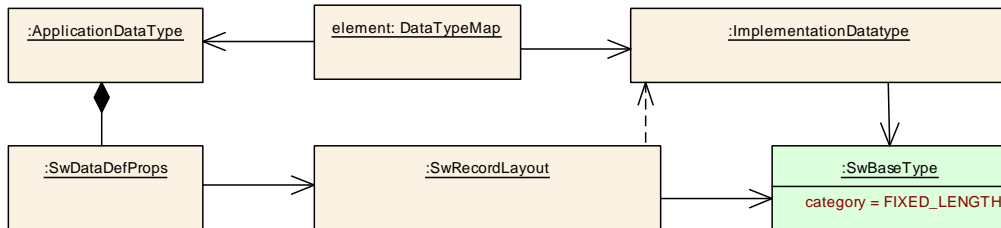


Figure 5.63: Dependency of **AutosarDataTypes** and **SwRecordLayouts**

[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**).] (**RS\_SWCT\_03215**)

Figures 5.64, 5.65, 5.66, 5.67, and 5.69 illustrate how data types can be derived from **SwRecordLayouts**. Please note that the figures simplify some aspects of the actual modeling. In particular, aggregations of **SwDataDefProps** are left out for the sake of visual clarity.

Note that in each of these diagrams, 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 primitive data types. The inner details of such entities are handled e.g. by service libraries.

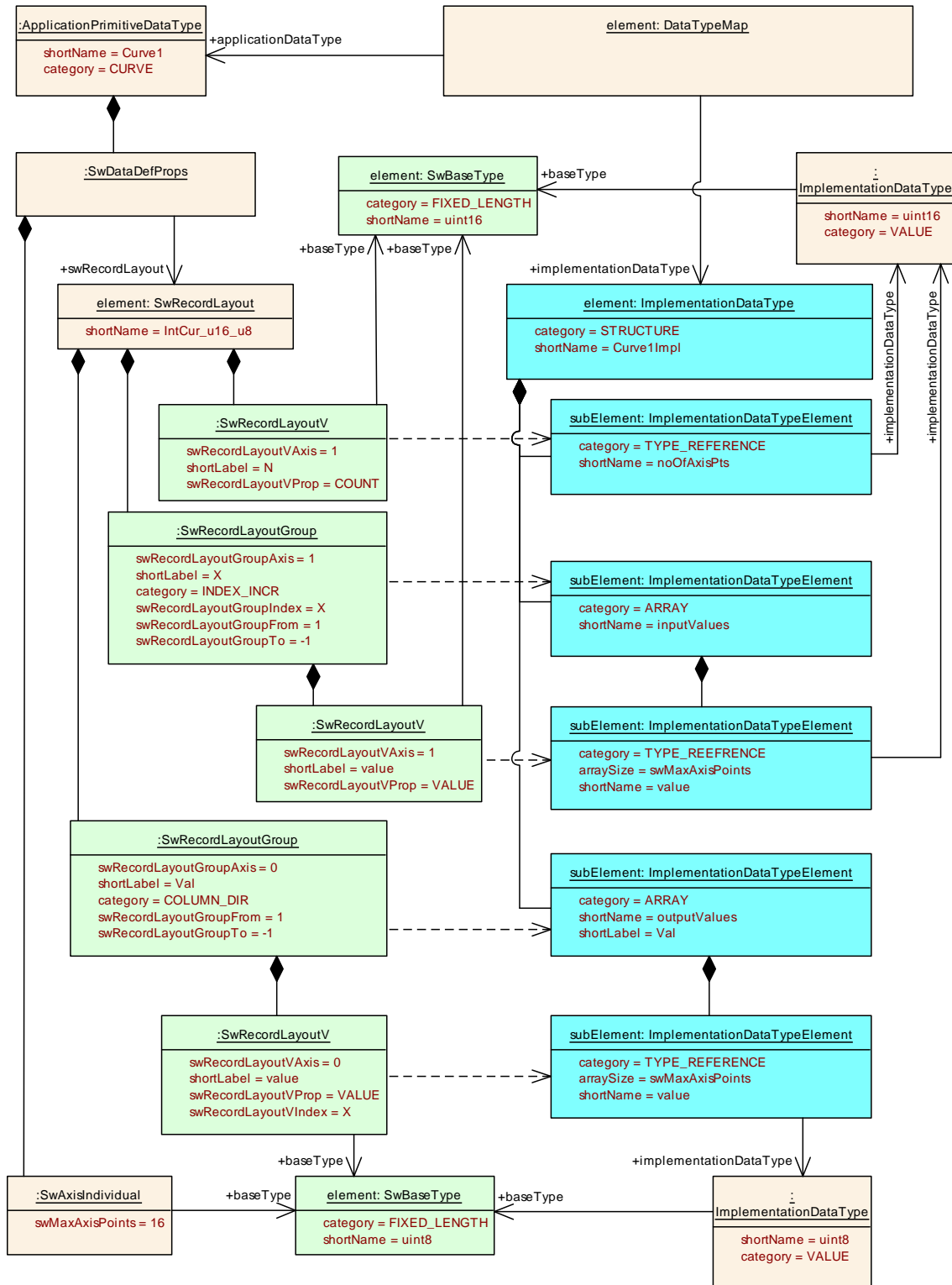
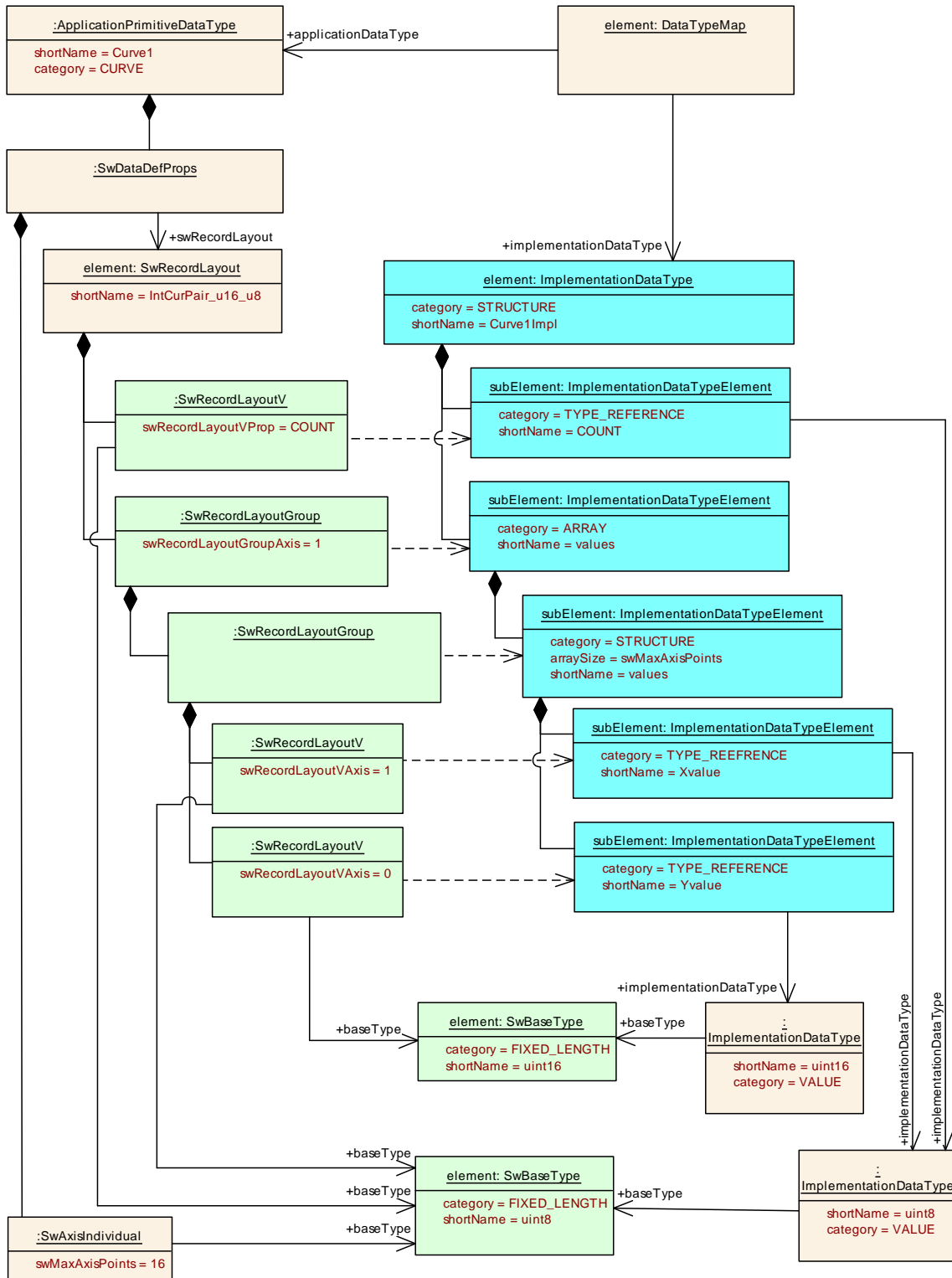
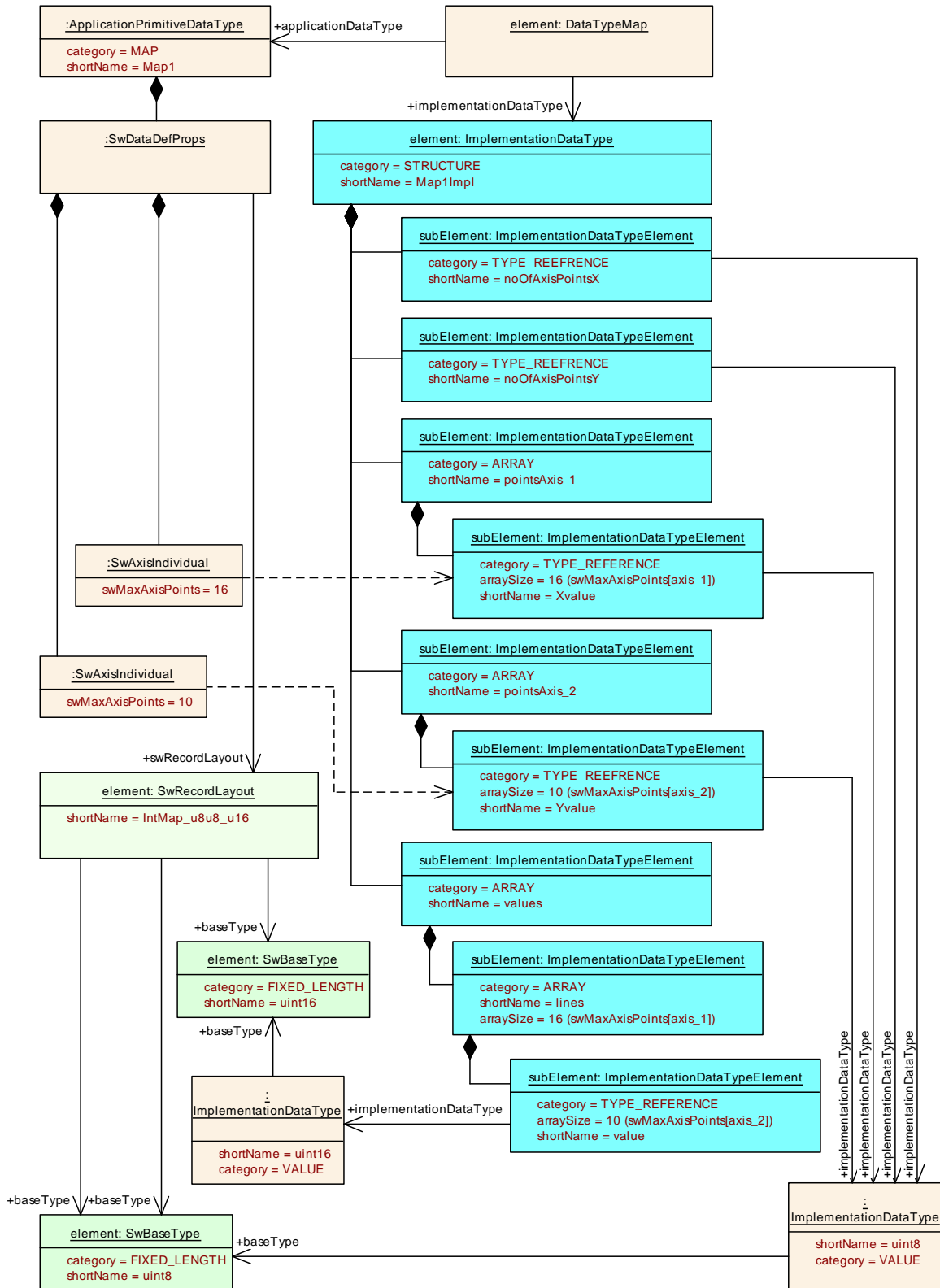


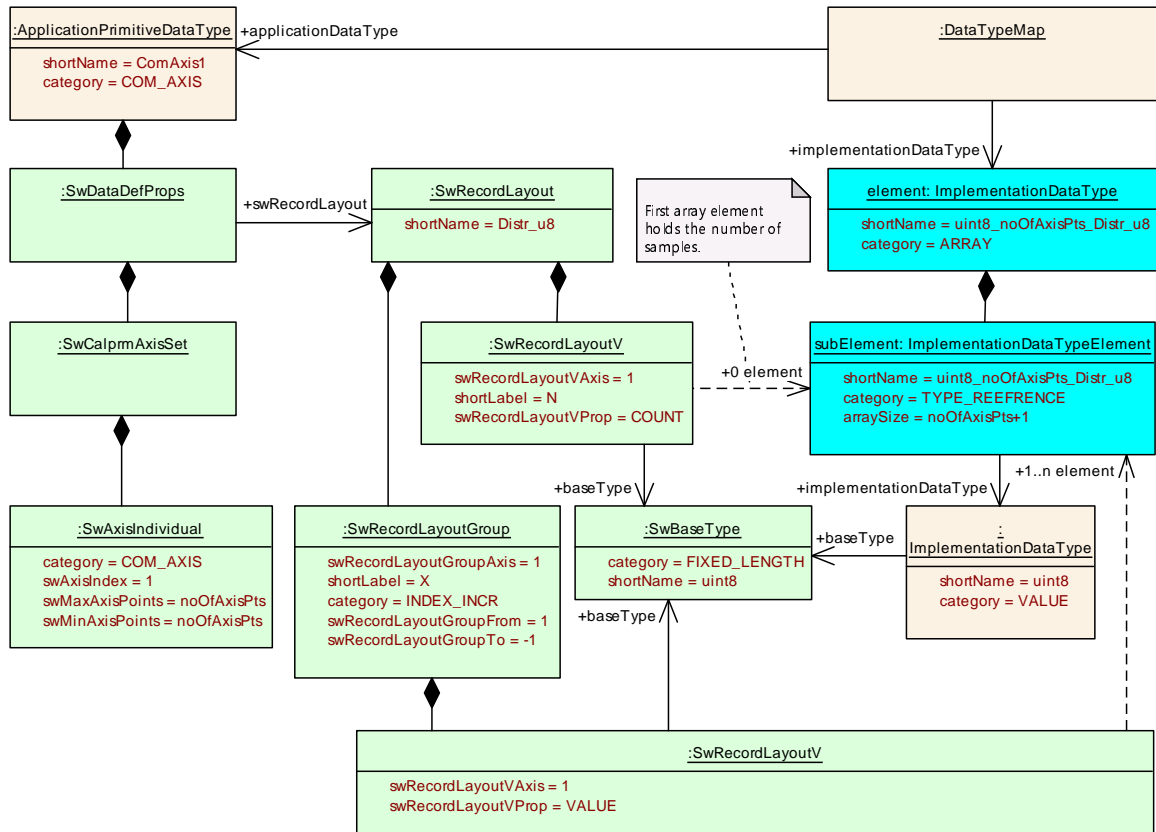
Figure 5.64: Curve implemented as two consecutive arrays



**Figure 5.65: Curve implemented as array of value pairs**



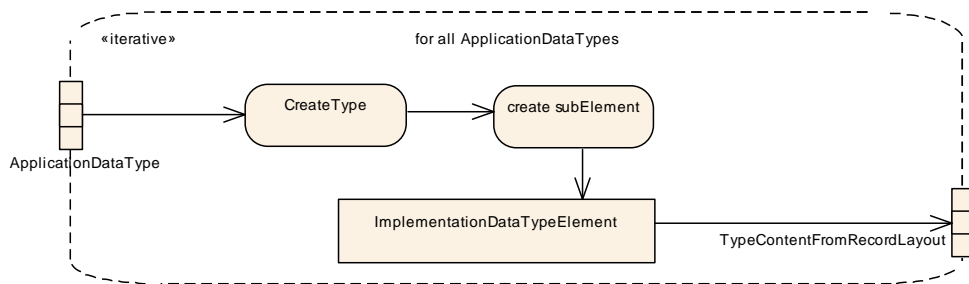
**Figure 5.66: Record layout and data type for a map**



**Figure 5.67: Record layout for the definition of a group axis**

The algorithm to generate the desired data types is illustrated in the following two diagrams.

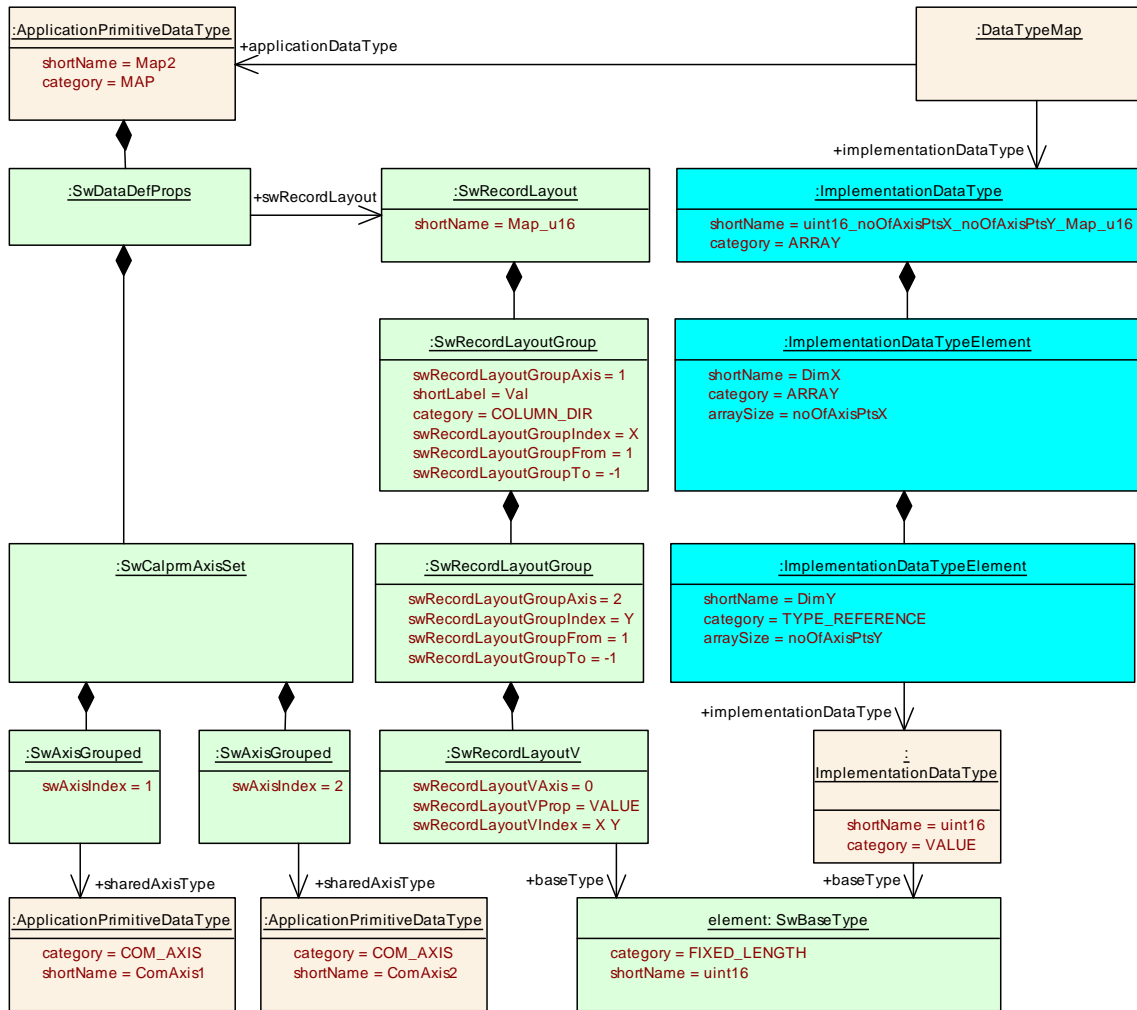
We create an `ImplementationDataType` for each `ApplicationDataType`. Figure 5.68 illustrates how to map the details.



**Figure 5.68: algorithm to map the details of an application data type to the corresponding implementation data type according to the record layout**

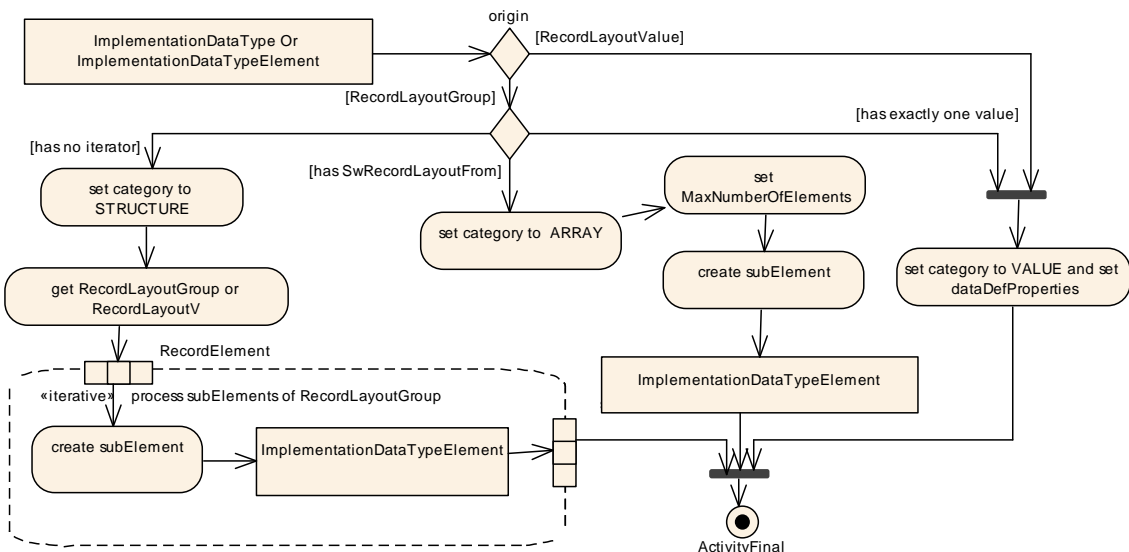
[TPS\_SWCT\_01299] Relation of `swRecordLayoutGroup` to `subElement` [For each `swRecordLayoutGroup` an appropriate `subElement` shall be created.

The algorithm shall be recursively applied to the newly created `ImplementationDataTypeElements`. As the record layout groups are nested, this recursion yields the complete structure in the `ImplementationDataType`.] (RS\_SWCT\_03215)



**Figure 5.69: Record layout for the definition of a map implemented by an array data type**

Please note that the refinement of the sub element happens according to the approach sketched in figure 5.70.



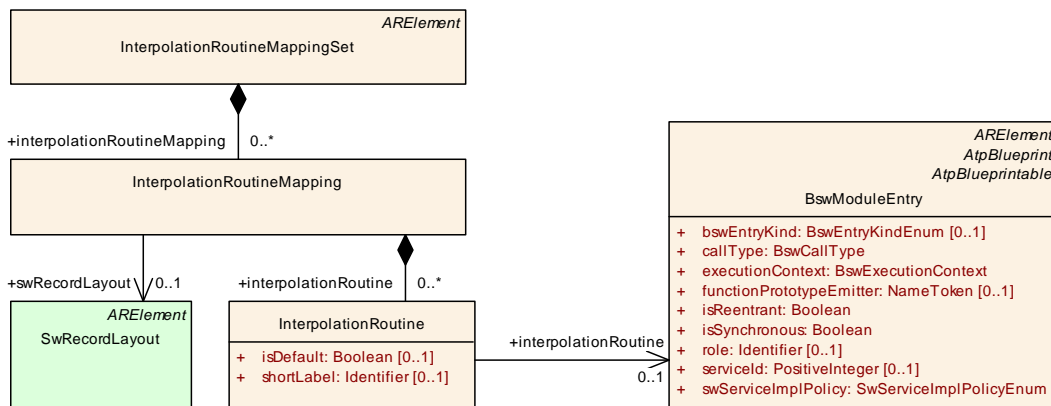
**Figure 5.70: refining subElements**

### 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 [SwDataDef-Props.swInterpolationMethod.](#) ([RS\\_SWCT\\_03215](#))



**Figure 5.71: Mapping of Record Layouts and Interpolation Routines**

<b>Class</b>	<b>InterpolationRoutineMappingSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::InterpolationRoutineMappingSet			
<b>Note</b>	This meta-class specifies a set of interpolation routine mappings. <b>Tags:</b> atp.recommendedPackage=InterpolationRoutineMappingSets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
interpolation Routine Mapping	<a href="#">InterpolationRoutineMapping</a>	*	aggr	This specifies one particular mapping of recordlayout and its matching interpolationRoutines.

**Table 5.115: InterpolationRoutineMappingSet**

<b>Class</b>	<b>InterpolationRoutineMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::InterpolationRoutineMappingSet			
<b>Note</b>	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.			





<b>Class</b>	<b>InterpolationRoutineMapping</b>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
interpolation Routine	<a href="#">InterpolationRoutine</a>	*	aggr	This is one particular interpolation routine which is mapped to the record layout.
swRecord Layout	<a href="#">SwRecordLayout</a>	0..1	ref	This refers to the record layout which is mapped to interpolation routines.

**Table 5.116: InterpolationRoutineMapping**

<b>Class</b>	<b>InterpolationRoutine</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::InterpolationRoutineMappingSet			
<b>Note</b>	This represents an interpolation routine taken to evaluate the contents of a curve or map against a specific input value.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
interpolation Routine	<a href="#">BswModuleEntry</a>	0..1	ref	This specifies a BswModuleEntry which implements the current interpolation method for the given record layout. <b>Tags:</b> xml.sequenceOffset=30
isDefault	Boolean	0..1	attr	This attribute specifies whether the enclosing InterpolationRoutine is considered the default in the context (defined by the System Template) of a given collection InterpolationRoutineMapping that owns the enclosing InterpolationRoutine. <b>Tags:</b> xml.sequenceOffset=20
shortLabel	Identifier	0..1	attr	This is the name of the interpolation method which is implemented by the referenced bswModuleEntry. It corresponds to swInterpolationMethod in SwDataDef Props. <b>Tags:</b> xml.sequenceOffset=10

**Table 5.117: InterpolationRoutine**

## 5.5.6 Display Presentation

### [TPS\_SWCT\_01756] Semantics of [SwDataDefProps.displayPresentation](#)

[The attribute [SwDataDefProps.displayPresentation](#) is used to control the presentation of data within measurement and calibration tools.

When such a tool displays a series of measurement values its useful to indicate to the displaying tool whether the series of measurement values can be seen as a continuous graph or as a set of discrete values, i.e. step-wise.

For instance, a continuous graph is appropriate for the case that the values do not bounce arbitrarily within one measurement cycle, e.g. a temperature variable.] ()

On the other hand, a discrete handling is correct if each value of the measured variable has a distinct meaning and therefore may arbitrarily change within one measurement cycle, e.g. a state variable.



Another use case is the indication of how an ECU utilizes a `DataPrototype` of category `CURVE`, `MAP`, or `CUBOID` to determine a single value out of one or several working points in axis.

This can be either done via interpolation between the sampling points on each axis or without interpolation by taking the nearest sampling point.

The first option requires the continuous representation for the determined value in the displaying tool whereas the second option expects a discrete handling of the determined value.

**[constr\_1592] Definition of `SwDataDefProps.displayPresentation` depending on the capabilities of the data type** [The definition of a `SwDataDefProps.displayPresentation` according to [constr\_1288] and [constr\_1289] shall only be applied for a `DataPrototype` of category `ARRAY` if the corresponding `ApplicationArrayDataType` or `ImplementationDataType` of category `ARRAY` supports the specification of a `SwDataDefProps.displayPresentation`.

This rule shall be imposed **at any time in the workflow.**]()

**[constr\_1602] Definition of `SwDataDefProps.displayPresentation` depending on the capabilities of the element** [The definition of a `SwDataDefProps.displayPresentation` according to [constr\_1007] and [constr\_1009] is only supported for an `ApplicationArrayDataType` or an `ImplementationDataType` of category `ARRAY` if the aggregated `ApplicationArrayDataType.element` or `ImplementationDataType.subElement` also supports the specification of a `SwDataDefProps.displayPresentation`.

This rule shall be imposed **at any time in the workflow.**]()

**[TPS\_SWCT\_01757] Not-applicable scenario for `presentationContinuous`** [If the semantics of the `DataPrototype` is described by means of a `CompuMethod` of category `TEXTTABLE`, `BITFIELD_TEXTTABLE` or `TAB_NOINTP` the option to set attribute `displayPresentation` is meaningless because the step-wise change of data is an intrinsic property of the data object.]()

**[TPS\_SWCT\_01758] Applicable value range of `SwDataDefProps.displayPresentation`** [If the semantics of a `DataPrototype` is described by means of a `CompuMethod` of category `IDENTICAL`, `LINEAR`, `RAT_FUNC` the attribute `SwDataDefProps.displayPresentation` describes the presentation of data for the complete value range.

If the semantics of a `DataPrototype` is described by means of a `CompuMethod` of category `SCALE_LINEAR_AND_TEXTTABLE` or `SCALE_RATIONAL_AND_TEXTTABLE` the attribute `SwDataDefProps.displayPresentation` describes the presentation of data only for the value range outside the `TEXTTABLE` values.]()

<b>Enumeration</b>	<b>DisplayPresentationEnum</b>
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties
<b>Note</b>	This meta-class represents the ability to provide values for controlling the presentation of data within measurement and calibration tools.
<b>Literal</b>	<b>Description</b>
presentation Continuous	The presentation of data shall form a continuous graph between data points. <b>Tags:</b> atp.EnumerationLiteralIndex=0
presentation Discrete	The presentation of data shall be step-shaped between data points. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 5.118: DisplayPresentationEnum**

## 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](#))

<b>Class</b>	<b>ConstantSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	Specification of a constant that can be part of a package, i.e. it can be defined stand-alone. <b>Tags:</b> atp.recommendedPackage=ConstantSpecifications			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
valueSpec	<a href="#">ValueSpecification</a>	0..1	aggr	Specification of an expression leading to a value for this constant.

**Table 5.119: ConstantSpecification**

**[constr\_1917] Existence of [ConstantSpecification.valueSpec](#)** [For each [ConstantSpecification](#), the aggregation of [ValueSpecification](#) in the role [valueSpec](#) shall exist **at the time when the contract phase generation is executed.**]()

<b>Class</b>	<b>ValueSpecification</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	Base class for expressions leading to a value which can be used to initialize a data object.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	<a href="#">AbstractRuleBasedValueSpecification</a> , <a href="#">ApplicationValueSpecification</a> , <a href="#">CompositeValueSpecification</a> , <a href="#">ConstantReference</a> , <a href="#">NotAvailableValueSpecification</a> , <a href="#">NumericalValueSpecification</a> , <a href="#">ReferenceValueSpecification</a> , <a href="#">TextValueSpecification</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
shortLabel	Identifier	0..1	attr	This can be used to identify particular value specifications for human readers, for example elements of a record type.

**Table 5.120: ValueSpecification**

<b>Class</b>	<b>CompositeValueSpecification</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	This abstract meta-class acts a base class for ValueSpecifications that have a composite form.			
<b>Base</b>	ARObject, <a href="#">ValueSpecification</a>			
<b>Subclasses</b>	<a href="#">ArrayValueSpecification</a> , <a href="#">RecordValueSpecification</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 5.121: CompositeValueSpecification**

<b>Class</b>	<b>ArrayValueSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	Specifies the values for an array.			
<b>Base</b>	ARObject, <a href="#">CompositeValueSpecification</a> , <a href="#">ValueSpecification</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
element (ordered)	<a href="#">ValueSpecification</a>	*	aggr	The value for a single array element. All Value Specifications aggregated by ArrayValueSpecification shall have the same structure.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
intendedPartial Initialization Count	PositiveInteger	0..1	attr	This attribute shall only have a meaning for dynamic arrays and shall be taken as a sanity check: the number filled in the attribute shall be identical to the number of ArrayValueSpecification.element.  If the attribute does not exist it means that no partial initialization is intended.

**Table 5.122: ArrayValueSpecification**

<b>Class</b>	<b>RecordValueSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	Specifies the values for a record.			
<b>Base</b>	ARObject, <a href="#">CompositeValueSpecification</a> , <a href="#">ValueSpecification</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	RecordValueSpecification			
field (ordered)	<a href="#">ValueSpecification</a>	*	aggr	<p>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 Value Specification independently of the shortNames.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>

**Table 5.123: RecordValueSpecification**

**[constr\_1918] Existence of [RecordValueSpecification.field](#)** [For each [RecordValueSpecification](#), the aggregation of [ValueSpecification](#) in the role `field` shall exist **at the time when the contract phase generation is executed.** ]()

Class	TextValueSpecification			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	The purpose of TextValueSpecification is to define the labels that correspond to enumeration values.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">ValueSpecification</a>			
Attribute	Type	Mult.	Kind	Note
value	VerbatimString	0..1	attr	<p>This is the value itself.</p> <p>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.</p>

**Table 5.124: TextValueSpecification**

**[constr\_1919] Existence of [TextValueSpecification.value](#)** [For each [TextValueSpecification](#), attribute `value` shall exist **at the time when the contract phase generation is executed.** ]()

Class	NumericalValueSpecification			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	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.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">ValueSpecification</a>			
Attribute	Type	Mult.	Kind	Note
value	Numerical	0..1	attr	<p>This is the value itself.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>

**Table 5.125: NumericalValueSpecification**

**[constr\_1920] Existence of [NumericalValueSpecification.value](#)** [For each [NumericalValueSpecification](#), attribute `value` shall exist **at the time when the contract phase generation is executed.** ]()

<b>Class</b>	<b>ReferenceValueSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	Specifies a reference to a data prototype to be used as an initial value for a pointer in the software.			
<b>Base</b>	ARObject, ValueSpecification			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
referenceValue	DataPrototype	0..1	ref	The referenced data prototype.

**Table 5.126: ReferenceValueSpecification**

**[constr\_1921] Existence of ReferenceValueSpecification.referenceValue**  
 [For each ReferenceValueSpecification, attribute referenceValue shall exist at the time when the contract phase generation is executed.]()

Figure 5.72 shows the specialized subclasses of ValueSpecification which allow defining values for different use cases.

**[TPS\_SWCT\_01178] Specialized subclasses of ValueSpecification** [The use case for specialized subclasses of ValueSpecification are:

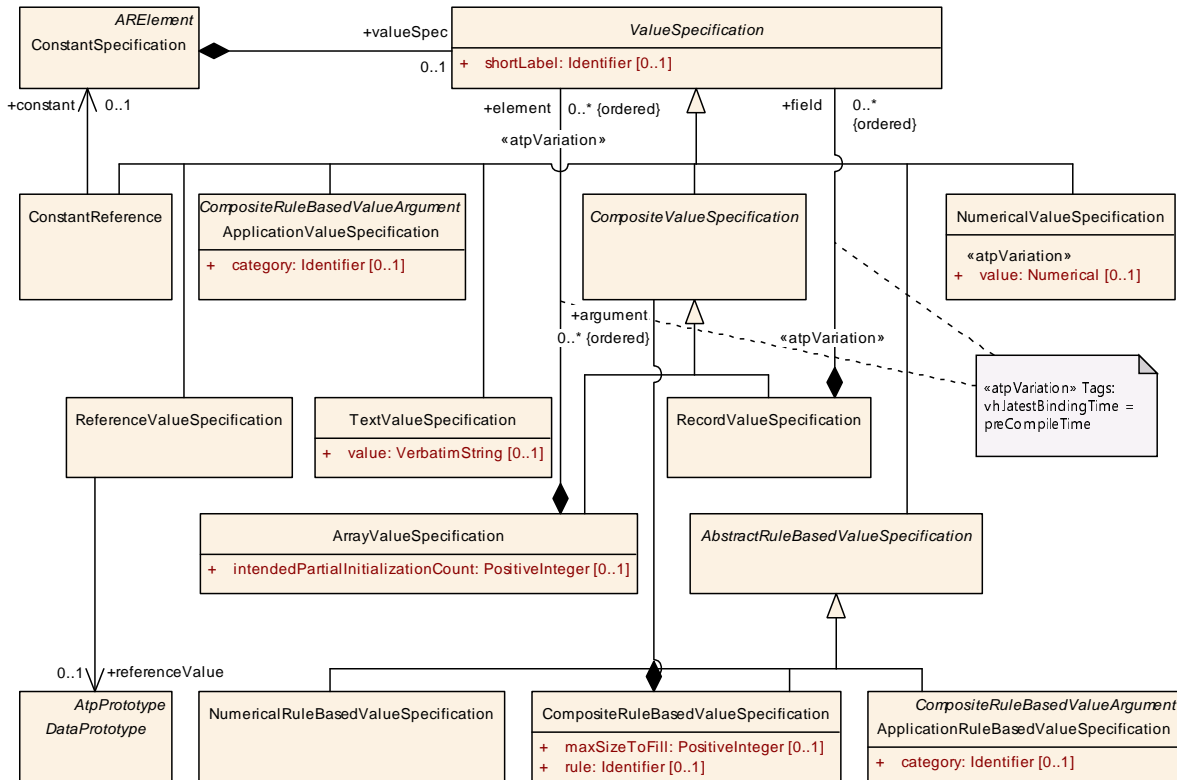
- Reference to a constant (which is actually a reusable value specification) by means of a ConstantReference.
- TextValueSpecification
- NumericalValueSpecification
- ArrayValueSpecification
- RecordValueSpecification
- ApplicationValueSpecification: this can be used to specify the value of Compound 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.

Note that ApplicationValueSpecification is modeled along the example of ASAM CDF (for more information please refer to [30]).

- 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 descriptions used to access calibration data for software emulation methods (see [6] for details).
- ApplicationRuleBasedValueSpecification, inside an ArrayValueSpecification, see [constr\_1779].
- NumericalRuleBasedValueSpecification, inside an ArrayValueSpecification, see [constr\_1779].
- CompositeRuleBasedValueSpecification, inside an ArrayValueSpecification, see [constr\_1779].

](RS\_SWCT\_03175)

It's 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 a `VariableDataPrototype/ParameterDataPrototype` typed by an `ApplicationPrimitiveDataType` of category `STRING` the `TextValueSpecification` actually has a different purpose (as defined by [constr\_1284]).



**Figure 5.72: Summary of ValueSpecification**

[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.swDataDef-Props.compuMethod` of category `TEXTTABLE` and `BITFIELD_TEXTTABLE`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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 [constr\_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 redundancy<sup>19</sup> in the meta-model since the intended data type of a given `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`:

**[TPS\_SWCT\_01838] `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.fields` shall be identical to the number of `ApplicationRecordDataType.elements`** [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.fields` shall be identical to the number of `ApplicationRecordDataType.elements`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1272] `RecordValueSpecification.fields` 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.fields` shall be identical to the number of `ImplementationDataType.subElements`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

If the corresponding `ApplicationRecordElement` is typed by an `ApplicationRecordDataType` then the comparison of structural compliance between `ApplicationRecordDataType` and `ValueSpecification` shall continue beyond the encountered `NotAvailableValueSpecification`.

---

<sup>19</sup>For example, “1” can be taken as a constant value for many data types. If the `ValueSpecification` were instead referring 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.

<b>Class</b>	<b>NotAvailableValueSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	This meta-class provides the ability to specify a ValueSpecification to state that the respective element is not available. This ability is needed to support the existence of ApplicationRecordElements where attribute isOptional ist set to the value True. <b>Tags:</b> atp.Status=draft			
<b>Base</b>	ARObject, ValueSpecification			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
defaultPattern	PositiveInteger	0..1	attr	The content of this attribute shall be used to initialize gaps in the memory occupied by a structured data type in the case that an NotAvailableValueSpecification is used. Note that this pattern is only applied during initialization!

**Table 5.127: NotAvailableValueSpecification**

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 [[TPS\\_SWCT\\_01838](#)] applies.

Please find more information about the creation of rule-based [ValueSpecifications](#) in section [5.6.6](#).

## 5.6.2 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](#).

<b>Class</b>	<b>ConstantReference</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	Instead of defining this value inline, a constant is referenced.			
<b>Base</b>	ARObject, ValueSpecification			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
constant	ConstantSpecification	0..1	ref	The referenced constant.

**Table 5.128: ConstantReference**

[[constr\\_1930](#)] Existence of [ConstantReference.constant](#) [For each [ConstantReference](#), attribute [constant](#) shall exist **at the time when the contract phase generation is executed.**]()



### 5.6.3 Values for Variable-Size Array

Variable-size data types have the ability to change the number of valid elements at run-time. However, in many situations it is necessary to define an `ArrayValueSpecification` for such a data type.

An `ArrayValueSpecification` that can be used for a *variable-size array* data type needs to be able to handle the following cases:

- Full initialization of the entire array-data type. This case is identical to the creation of a `ArrayValueSpecification` for a fixed-size array data type.
- Provision of values for the **first n elements** of the *variable-size array*. This case is also known as *partial initialization*.
- Creation of an empty `ArrayValueSpecification`, i.e. an `ArrayValueSpecification` carries the semantics of intentionally initializing 0 elements of a *variable-size array*. Note the semantical difference between not initializing at all and intentionally initializing 0 elements of the *variable-size array*.

All the described cases shall be supported by AUTOSAR. As already described, the existence of an `ArrayValueSpecification` with the full number of elements is identical to the fixed size case. The "empty" case could be seen as a subset of the partial initialization.

The partial initialization of *variable-size arrays* has two facets:

**[TPS\_SWCT\_01793] Initialization of a *variable-size array* typed by an `ImplementationDataType`** [A *variable-size array* that is modeled by means of an `ImplementationDataType` is actually existing as a structure consisting of a *size indicator* and an *array that carries the payload*.

Therefore, the *partial initialization* shall be implemented by explicitly initializing the *size indicator* to a value between 0 and the applicable `ImplementationDataTypeElement.arraySize` and provide the corresponding number of `ValueSpecifications` for the *payload*.] ([RS\\_SWCT\\_03175](#), [RS\\_SWCT\\_03181](#))

**[TPS\_SWCT\_01794] Initialization of a *variable-size array* typed by an `ApplicationArrayDataType`** [A *variable-size array* that is modeled by means of an `ApplicationArrayDataType` where attribute `arraySizeSemantics` is set to `variableSize` does not contain any size-indicator element and therefore requires a different approach for *partial initialization*.

For this purpose, `ArrayValueSpecification.intendedPartialInitializationCount` shall be used for the specification of the number of elements that shall be initialized.]([RS\\_SWCT\\_03175](#), [RS\\_SWCT\\_03181](#))

The applicability of attribute `ArrayValueSpecification.intendedPartialInitializationCount` is limited to the use case of initializing a *variable-size array* typed by an `ApplicationArrayDataType`. AUTOSAR does not foresee any other use case for this attribute.

**[constr\_1712] Existence of attribute `ArrayValueSpecification.intendedPartialInitializationCount`** [An `ArrayValueSpecification` where attribute `intendedPartialInitializationCount` exists shall only be applied for the initialization of an `ApplicationArrayDataType` where attribute `arraySizeSemantics` is set to `variableSize`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1273] Rules for the initialization of `ApplicationArrayDataType` by means of `ArrayValueSpecification`** [The following rules apply for the initialization of a `DataPrototype` typed by an `ApplicationArrayDataType` by means of an `ArrayValueSpecification`:

- If the attribute `ApplicationArrayDataType.element.arraySizeSemantics` is set to **fixedSize** then the `ArrayValueSpecification` shall exactly match the structure of the `ApplicationArrayDataType`.

This means that the number of `ArrayValueSpecification.elements` shall be identical to the value of `ApplicationArrayDataType.element.maxNumberOfElements`.

- If the attribute `ApplicationArrayDataType.element.arraySizeSemantics` is set to **variableSize** and the `ArrayValueSpecification` **does not define** attribute `intendedPartialInitializationCount` then `ArrayValueSpecification` shall **exactly** match the structure of the `ApplicationArrayDataType`.

This means that the number of `ArrayValueSpecification.elements` shall be identical to the value of `ApplicationArrayDataType.element.maxNumberOfElements`.

- If the attribute `ApplicationArrayDataType.element.arraySizeSemantics` is set to **variableSize** and the `ArrayValueSpecification` specifies a value for attribute `intendedPartialInitializationCount` then `ArrayValueSpecification` shall contain **exactly** `intendedPartialInitializationCount` elements.

This includes the case that the value of `intendedPartialInitializationCount` is set to 0 (i.e. "empty" initialization) and the case that the `intendedPartialInitializationCount` is set to the value of the respective `ApplicationArrayElement.maxNumberOfElements` (i.e. "full" initialization).

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1274] Rules for the initialization of array-shaped `ImplementationDataType` with a fixed size by means of `ArrayValueSpecification`** [The following rule applies for the initialization of a `DataPrototype` typed by an `ImplementationDataType` of category ARRAY where attribute `ImplementationDataType`.

`subElement.arraySizeSemantics` is set to `fixedSize` by means of an `ArrayValueSpecification`: the `ArrayValueSpecification` shall exactly match the structure of the `ImplementationDataType`.

This means that the number of `ArrayValueSpecification.elements` shall be identical to the value of `ImplementationDataType.subElement.arraySize`.

This rule shall be imposed **at the time when the contract phase generation is executed.** `()`

Please note that the initialization of an `ImplementationDataType` that represents a *variable-size array* is clarified in [\[TPS\\_SWCT\\_01793\]](#).

More details can be found in section [5.6.6](#).

### 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`, `swValueBlockSizeMult` 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 `initValues` matches the range of the involved system constants.

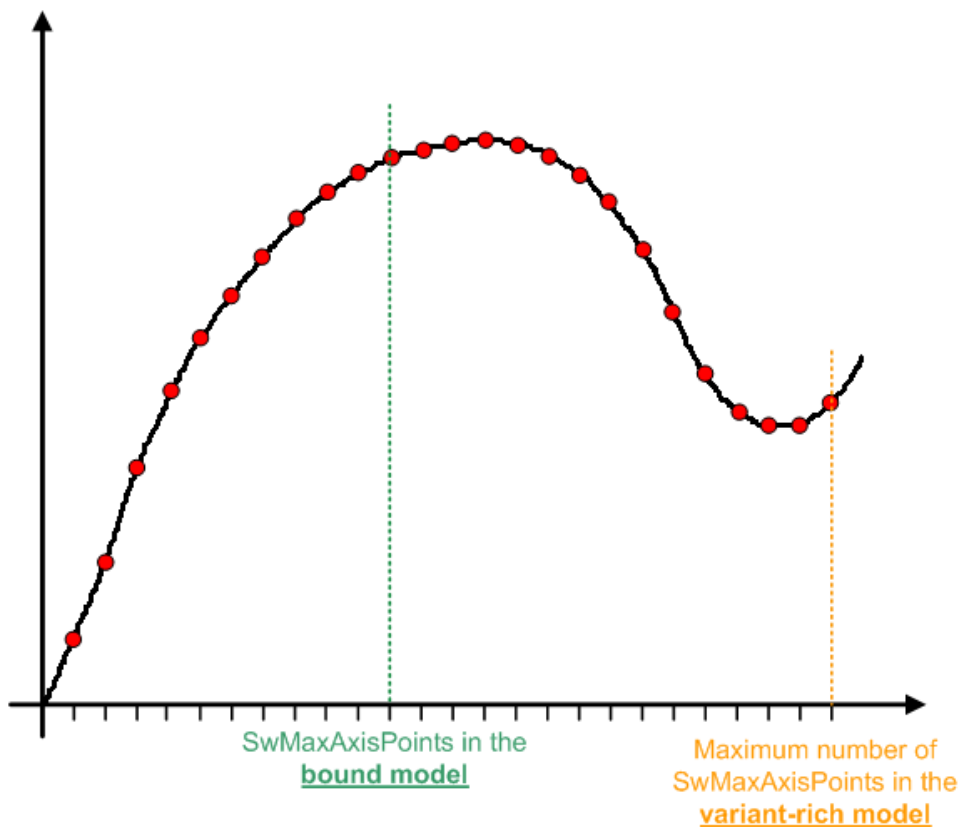
<b>Class</b>	<b>SwSystemconst</b>			
<b>Package</b>	M2::MSR::DataDictionary::SystemConstant			
<b>Note</b>	<p>This element defines a system constant which serves an input to select a particular variation point. In particular a system constant serves as an operand of the binding function (<code>swSyscond</code>) in a Variation point.</p> <p>Note that the binding process can only happen if a value was assigned to to the referenced system constants.</p> <p><b>Tags:</b><code>atp.recommendedPackage=SwSystemconst</code></p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpDefinition</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
<code>swDataDef Props</code>	<a href="#">SwDataDefProps</a>	0..1	aggr	<p>This denotes the data definition properties of the system constant. This supports to express the limits and optionally a conversion within the internal to physical values by a <code>compu</code> method.</p> <p><b>Tags:</b><code>xml.sequenceOffset=40</code></p>

**Table 5.129: SwSystemconst**

**[TPS\_SWCT\_01839] 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`.] ([RS\\_SWCT\\_03216](#), [RS\\_SWCT\\_03148](#))

**[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 5.73.



**Figure 5.73: Explanation of `swMaxAxisPoints`**

**[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.

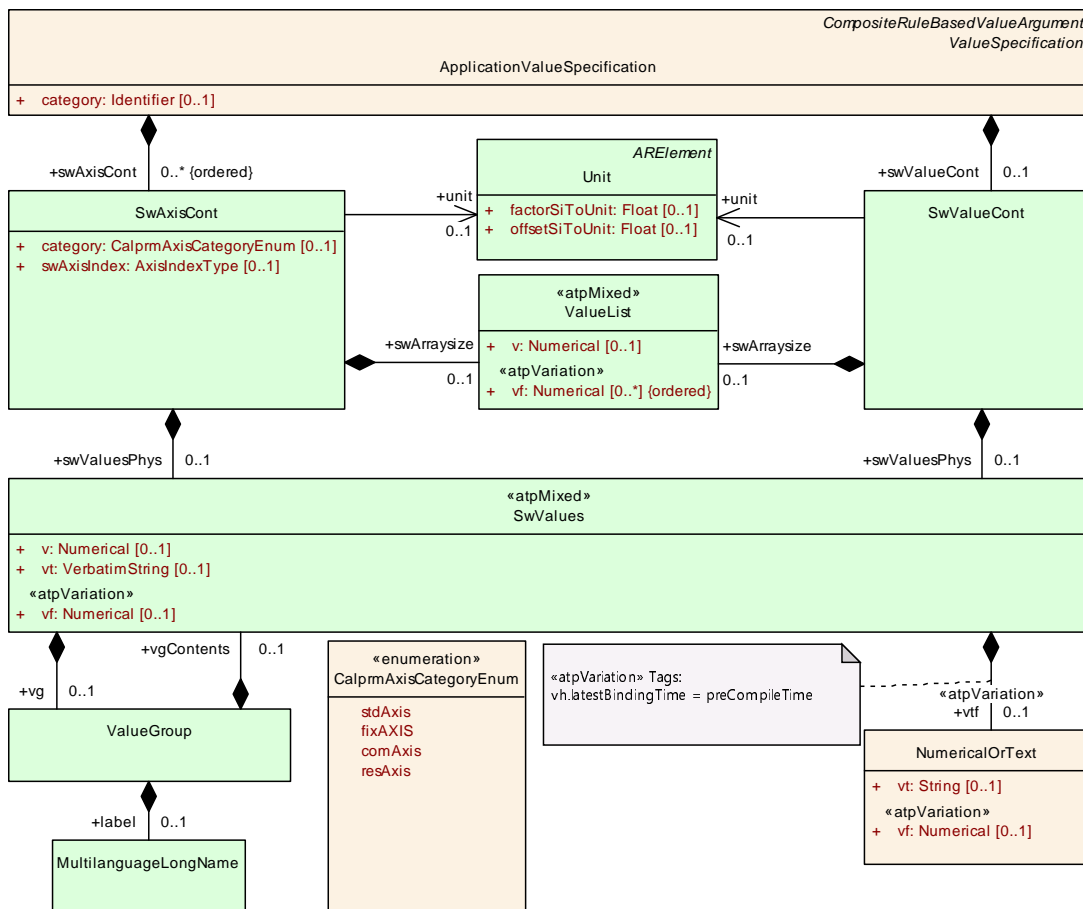
This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2051] Mandatory information of a SwValueCont** [If the attribute `swValueCont` is defined for an `ApplicationValueSpecification` the `SwValueCont`

shall always define the attribute `swArraysSize` if the `ApplicationValueSpecification` is of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, `RES_AXIS`, or `VAL_BLK`.

This rule shall be imposed **at the time when the contract phase generation is executed.** `()`

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_2050]` and `[constr_2051]` shall support a consistent handling of single and multidimensional `Compound Primitive Data Types`.



**Figure 5.74: Definition of an `ApplicationValueSpecification`**

**[constr\_2052] Values of `swArraysSize` and the number of values provided by `swValuesPhys` shall be consistent.** `[swValuesPhys]` shall define as many values as the attribute `swArraysSize` (if this attribute exists) 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 `swArraysSize` values are provided, the values have to be multiplied in order to get the total number of `swValuesPhys` values.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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 Data Type** defines.

If the **ApplicationValueSpecification** defines values for a **Compound Primitive Data Type** with more than one input axis the `swArraysizes` 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 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**.

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**.]()

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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Attribute of ApplicationValueSpecification	Attribute Existence per Category									
	VALUE	STRING	BOOLEAN	COM_AXIS	RES_AXIS	CURVE	MAP	CUBOID	CUBE_4	CUBE_5
<code>swValueCont</code>	D	D	D	D	D	D	D	D	D	D
<code>swValueCont.unit</code>	O	O	O	O	O	O	O	O	O	O
<code>swValueCont.swValuesPhys</code>	D	D	D	D	D	D	D	D	D	D
<code>swValueCont.swArraysizes</code>	N/A	N/A	N/A	D	D	D	D	D	D	D
<code>swAxisCont</code>	N/A	N/A	N/A	N/A	D	D	D	D	D	D
<code>swAxisCont.unit</code>	N/A	N/A	N/A	N/A	O	O	O	O	O	O
<code>swAxisCont.category</code>	N/A	N/A	N/A	N/A	D	D	D	D	D	D
<code>swAxisCont.swAxisIndex</code>	N/A	N/A	N/A	N/A	D	D	D	D	D	D
<code>swAxisCont.swArraysizes</code>	N/A	N/A	N/A	N/A	D	D	D	D	D	D
<code>swAxisCont.swValuesPhys</code>	N/A	N/A	N/A	N/A	D	O(1)	O(1)	O(1)	O(1)	O(1)

**Table 5.130: Allowed Attributes vs. category for ApplicationValueSpecification**

The following legend applies to the cells in table 5.130:

**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 5.81:

(1) Optional if `COM_AXIS` or `RES_AXIS` is used, otherwise attribute shall exist.

**[constr\_1519] Existence of attributes vs. category of ApplicationValueSpecification** [The existence of attributes of meta-class `ApplicationValueSpecification` vs. the value of `category` is regulated by Table 5.130.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

<b>Class</b>	<b>ApplicationValueSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>CompositeRuleBasedValueArgument</i> , <i>ValueSpecification</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
category	Identifier	0..1	attr	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)	<a href="#">SwAxisCont</a>	*	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	<a href="#">SwValueCont</a>	0..1	aggr	This represents the values of a Compound Primitive Data Type.

**Table 5.131: ApplicationValueSpecification**

<b>Class</b>	<b>SwAxisCont</b>			
<b>Package</b>	M2::MSR::CalibrationData::CalibrationValue			
<b>Note</b>	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 <code>COM_AXIS</code> , <code>RES_AXIS</code> are for display only. For editing and processing, only the values in the related GroupAxis are binding.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	SwAxisCont			
category	<a href="#">CalprmAxisCategory Enum</a>	0..1	attr	This category specifies the particular axis types: <ul style="list-style-type: none"> <li>• STD_AXIS</li> <li>• COM_AXIS</li> <li>• RES_AXIS (swArraysize necessary)</li> </ul> <b>Tags:</b> xml.sequenceOffset=20
swArraysize	<a href="#">ValueList</a>	0..1	aggr	For multidimensional compound primitives (curve, map ...) it is necessary to know the dimensions. They are specified using swArraySize. <ul style="list-style-type: none"> <li>• RES_AXIS</li> </ul> <b>Tags:</b> xml.sequenceOffset=70
swAxisIndex	<a href="#">AxisIndexType</a>	0..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. <b>Tags:</b> xml.sequenceOffset=50
swValuesPhys	<a href="#">SwValues</a>	0..1	aggr	swValuesPhys represents the values in the physical domain. <b>Tags:</b> xml.sequenceOffset=80
unit	<a href="#">Unit</a>	0..1	ref	This represents the physical unit of the provided values. <b>Tags:</b> xml.sequenceOffset=30
unitDisplay Name	<a href="#">SingleLanguageUnit Names</a>	0..1	aggr	This represents the display name which is used for the physical unit of the axis. <b>Tags:</b> xml.sequenceOffset=40

**Table 5.132: SwAxisCont**

**[constr\_10017] Existence of attribute [SwAxisCont.category](#)** [For each [SwAxisCont](#), attribute [category](#) shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_10040] Value of [ApplicationValueSpecification.swAxisCont.category](#)** [The value of attribute [ApplicationValueSpecification.swAxisCont.category](#) shall not be set to [fixAXIS](#) **at the time when the contract phase generation is executed.**]()

Rationale for the existence of [\[constr\\_10040\]](#): the value [fixAXIS](#) for [category](#) indicates that the respective axis that is calculated out of the value of calibration parameters. Obviously, it does not make sense to initialize an axis of this kind and therefore it is excluded from the usage inside an [ApplicationValueSpecification](#).

**[constr\_10018] Existence of attribute [SwAxisCont.swAxisIndex](#)** [For each [SwAxisCont](#), attribute [swAxisIndex](#) shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_10019] Existence of attribute [SwAxisCont.swValuesPhys](#)** [For each [SwAxisCont](#), attribute [swValuesPhys](#) shall exist **at the time when the contract phase generation is executed.**]()



<b>Class</b>	<b>SwValueCont</b>			
<b>Package</b>	M2::MSR::CalibrationData::CalibrationValue			
<b>Note</b>	This metaclass represents the content of one particular SwInstance.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swArraysSize	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, VAL_BLK.  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.  <b>Tags:</b> xml.sequenceOffset=40
swValuesPhys	SwValues	0..1	aggr	swValuesPhys represents the values in the physical domain.  <b>Tags:</b> xml.sequenceOffset=50
unit	Unit	0..1	ref	This represents the physical unit of the provided values.  <b>Tags:</b> xml.sequenceOffset=20
unitDisplay Name	SingleLanguageUnit Names	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.  <b>Tags:</b> xml.sequenceOffset=30

**Table 5.133: SwValueCont**

<b>Class</b>	<<atpMixed>> <b>SwValues</b>			
<b>Package</b>	M2::MSR::CalibrationData::CalibrationValue			
<b>Note</b>	<p>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).</p> <p>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 Value Group. But the processing needs to be done as if vg is not there.</p> <p>Note that numerical values and textual values should not be mixed.</p>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
v	Numerical	0..1	attr	This is a non variant Value. It is provided for sake of Compatibility to ASAM CDF.  <b>Tags:</b> xml.sequenceOffset=40
vf	Numerical	0..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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
vg	ValueGroup	0..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.  <b>Tags:</b> xml.sequenceOffset=50





<b>Class</b>		<<atpMixed>> <b>SwValues</b>		
vt	VerbatimString	0..1	attr	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. <b>Tags:</b> xml.sequenceOffset=30
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.  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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 5.134: SwValues**

<b>Class</b>		<b>ValueGroup</b>		
<b>Package</b>		M2::MSR::CalibrationData::CalibrationValue		
<b>Note</b>		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.		
<b>Base</b>		ARObject		
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
label	MultilanguageLong Name	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. <b>Tags:</b> xml.sequenceOffset=20
vgContents	SwValues	0..1	aggr	This represents the contents of the value group. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

**Table 5.135: ValueGroup**

<b>Class</b>		<<atpMixed>> <b>ValueList</b>		
<b>Package</b>		M2::MSR::DataDictionary::DataDefProperties		
<b>Note</b>		This is a generic list of numerical values.		
<b>Base</b>		ARObject		
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
v	Numerical	0..1	attr	This is a particular numerical value without variation. <b>Tags:</b> xml.sequenceOffset=30





<b>Class</b>	<<atpMixed>> ValueList			
vf (ordered)	Numerical	*	attr	This is one entry in the list of numerical values <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.roleElement=true xml.roleWrapperElement=false xml.typeElement=false xml.typeWrapperElement=false

**Table 5.136: ValueList**

<b>Class</b>	NumericalOrText			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
vt	String	0..1	attr	This attribute represents the ability to provide a textual value. <b>Tags:</b> xml.sequenceOffset=20

**Table 5.137: NumericalOrText**

### 5.6.5 Values for BITFIELD\_TEXTTABLE

A [CompuMethod](#) of [category BITFIELD\\_TEXTTABLE](#) effectively defines a data type where single bits or a set of single bits have an individual meaning.

In other words, the definition of a [ValueSpecification](#) according to a [CompuMethod](#) of [category BITFIELD\\_TEXTTABLE](#) translates into the statement: “each unique value of attribute [CompuScale.mask](#) (because this is how the single bits or sets of single bits are identified) in the context of the enclosing [CompuMethod](#) delivers one contribution to the definition of the [ValueSpecification](#)”.

**[TPS\_SWCT\_01792] Initialization of a [DataPrototype](#) associated with a [CompuMethod](#) of [category BITFIELD\\_TEXTTABLE](#)** [The initialization of a [DataPrototype](#) associated with a [CompuMethod](#) of [category BITFIELD\\_TEXTTABLE](#) requires the simultaneous initialization of all single bits or sets of bits represented by the respective definition of bitmask.

Access to the elements is possible by using bit-operations, therefore the initialization needs to be defined in a way that bit-operations can be used to apply the provided value.

In other words, the overall value shall be created out of bitwise or-ing (represented by the usage of the “|” symbol) contributions from the individual subsets.]()

Listing 5.20 shows the definition of an [ApplicationValueSpecification](#) that can be taken e.g. for the initialization of a [DataPrototype](#) typed by an [Application-DataType](#) that refers to a [CompuMethod](#) of category `BITFIELD_TEXTTABLE`.

**Listing 5.20: Example for the definition of an [ApplicationValueSpecification](#) for `BITFIELD_TEXTTABLE`**

```
<APPLICATION-VALUE-SPECIFICATION>
  <SHORT-LABEL>tirePressureInit</SHORT-LABEL>
  <CATEGORY>VALUE</CATEGORY>
  <SW-VALUE-CONT>
    <SW-VALUES-PHYS>
      <VT>problem_low_Pressure|rearRight_yes|rearLeft_yes|frontRight_no|
        frontLeft_no</VT>
    </SW-VALUES-PHYS>
  </SW-VALUE-CONT>
</APPLICATION-VALUE-SPECIFICATION>
```

Listing 5.21 shows the definition of an [TextValueSpecification](#) that can be taken e.g. for the initialization of a [DataPrototype](#) typed by an [Implementation-DataType](#) that refers to a [CompuMethod](#) of category `BITFIELD_TEXTTABLE`.

**Listing 5.21: Example for the definition of a [TextValueSpecification](#) for `BITFIELD_TEXTTABLE`**

```
<TEXT-VALUE-SPECIFICATION>
  <SHORT-LABEL>tirePressureInit</SHORT-LABEL>
  <VALUE>problem_low_Pressure|rearRight_yes|rearLeft_yes|frontRight_no|
    frontLeft_no</VALUE>
</TEXT-VALUE-SPECIFICATION>
```

Listing 5.22 shows the definition of an [NumericalValueSpecification](#) that can be taken e.g. for the initialization of a [DataPrototype](#) typed by an [Implementation-DataType](#) that refers to a [CompuMethod](#) of category `BITFIELD_TEXTTABLE`.

**Listing 5.22: Example for the definition of a [NumericalValueSpecification](#) for `BITFIELD_TEXTTABLE`**

```
<NUMERICAL-VALUE-SPECIFICATION>
  <SHORT-LABEL>tirePressureInit</SHORT-LABEL>
  <VALUE>42</VALUE>
</NUMERICAL-VALUE-SPECIFICATION>
```

Please note that areas of the initialized [DataPrototype](#) that are not covered by the bit-masks may contain arbitrary values. If this needs to be avoided, it is necessary to add “dummy” bit-masks to the semantically not relevant parts of the value of the [DataPrototype](#).

## 5.6.6 Specification of Values based on Rules

### 5.6.6.1 Support for primitive Data Types

#### [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.

Please note that [ApplicationRuleBasedValueSpecification](#) applies for arrays typed by a primitive data type. Rule-based value specification of arrays of a composite data type is done by means of the [CompositeRuleBasedValueSpecification](#).

<b>Class</b>	<a href="#">AbstractRuleBasedValueSpecification</a> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	This represents an abstract base class for all rule-based value specifications.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">ValueSpecification</a>			
<b>Subclasses</b>	<a href="#">ApplicationRuleBasedValueSpecification</a> , <a href="#">CompositeRuleBasedValueSpecification</a> , <a href="#">NumericalRuleBasedValueSpecification</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

Table 5.138: [AbstractRuleBasedValueSpecification](#)

#### [constr\_1779] Scope of the definition of an [AbstractRuleBasedValueSpecification](#)

[An [AbstractRuleBasedValueSpecification](#) shall only be defined in the context of an [ArrayValueSpecification](#) or a [ConstantSpecification](#). If the [AbstractRuleBasedValueSpecification](#) is defined in the context of a [ConstantSpecification](#) then a reference to this [ConstantSpecification](#) shall only be created in the context of an [ArrayValueSpecification](#).

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

<b>Class</b>	<a href="#">ApplicationRuleBasedValueSpecification</a>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants





<b>Class</b>		<b>ApplicationRuleBasedValueSpecification</b>		
<b>Note</b>	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).			
<b>Base</b>	<i>ARObject</i> , <i>AbstractRuleBasedValueSpecification</i> , <i>CompositeRuleBasedValueArgument</i> , <i>ValueSpecification</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
category	Identifier	0..1	attr	This represents the category of the RuleBasedValue Specification <b>Tags:</b> xml.sequenceOffset=-20
swAxisCont (ordered)	<a href="#">RuleBasedAxisCont</a>	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second sw AxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	<a href="#">RuleBasedValueCont</a>	0..1	aggr	This represents the values of an array or Compound Primitive Data Type.

**Table 5.139: ApplicationRuleBasedValueSpecification**

**[constr\_1922] Existence of [ApplicationRuleBasedValueSpecification.category](#)** [For each [ApplicationRuleBasedValueSpecification](#), attribute [category](#) shall exist **at the time when the RTE is generated.**]()

Please note that attribute [ApplicationRuleBasedValueSpecification.category](#) is mentioned in [\[constr\\_2058\]](#), which would fail if the attribute does not exist.

**[constr\_10041] Value of [ApplicationRuleBasedValueSpecification.swAxisCont.category](#)** [The value of [ApplicationValueSpecification.swAxisCont.category](#) shall not be set to `fixAXIS`]()

Rationale for the existence of [\[constr\\_10041\]](#): the value `fixAXIS` of attribute [category](#) indicates that the respective axis that is calculated out of the value of calibration parameters. Obviously, it does not make sense to initialize an axis of this kind and therefore it is excluded from the usage inside an [ApplicationRuleBasedValueSpecification](#).

<b>Class</b>		<b>RuleBasedAxisCont</b>		
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	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 COM_AXIS, RES_AXIS are for display only. For editing and processing, only the values in the related GroupAxis are binding.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	RuleBasedAxisCont			
category	<a href="#">CalprmAxisCategory Enum</a>	0..1	attr	This category specifies the particular axis types: <ul style="list-style-type: none"> <li>• STD_AXIS</li> <li>• COM_AXIS</li> <li>• RES_AXIS (swArraysSize necessary)</li> </ul> <b>Tags:</b> xml.sequenceOffset=20
ruleBased Values	<a href="#">RuleBasedValue Specification</a>	0..1	aggr	This represents the rule based value specification for the axis of a compound primitive (curve, map). <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=80 xml.typeWrapperElement=false
swArraysSize	<a href="#">ValueList</a>	0..1	aggr	For multidimensional compound primitives (curve, map ...) it is necessary to know the dimensions. They are specified using swArraySize. <b>Tags:</b> xml.sequenceOffset=40
swAxisIndex	<a href="#">AxisIndexType</a>	0..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. <b>Tags:</b> xml.sequenceOffset=50
unit	<a href="#">Unit</a>	0..1	ref	This represents the physical unit of the provided values. <b>Tags:</b> xml.sequenceOffset=30

**Table 5.140: RuleBasedAxisCont**

**[constr\_1923] Existence of [RuleBasedAxisCont.ruleBasedValues](#)** [For each [RuleBasedAxisCont](#), attribute [ruleBasedValues](#) shall exist at the time when the contract phase generation is executed.] ()

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	Type	Mult.	Kind	Note
ruleBased Values	<a href="#">RuleBasedValue Specification</a>	0..1	aggr	This represents the rule based value specification for the array or compound primitive (CURVE, MAP, CUBOID, CUBE_4, CUBE_5, VAL_BLK). <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=80 xml.typeWrapperElement=false
swArraysSize	<a href="#">ValueList</a>	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, VAL_BLK. 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. <b>Tags:</b> xml.sequenceOffset=40





Class	RuleBasedValueCont			
unit	<a href="#">Unit</a>	0..1	ref	This represents the physical unit of the provided values. <b>Tags:</b> xml.sequenceOffset=30

**Table 5.141: RuleBasedValueCont**

**[constr\_1924] Existence of [RuleBasedValueCont.ruleBasedValues](#)** [For each [RuleBasedValueCont](#), attribute [ruleBasedValues](#) shall exist **at the time when the contract phase generation is executed.**]()

In case the [ApplicationRuleBasedValueSpecification](#) is applied to [Compound Primitive Data Types](#) basically the same rules apply for [ApplicationRuleBasedValueSpecification](#) 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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_2058] Mandatory information of a [RuleBasedValueCont](#)** [If the attribute [swValueCont](#) is defined for an [ApplicationRuleBasedValueSpecification](#) the [RuleBasedValueCont](#) shall always define the attribute [swArraysize](#) if the [ApplicationRuleBasedValueSpecification](#) is of category CURVE, MAP, CUBOID, CUBE\_4, CUBE\_5, COM\_AXIS, RES\_AXIS, or VAL\_BLK.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Please note that the definition of attribute [swArraysize](#) is not required for an [ApplicationRuleBasedValueSpecification](#) of category ARRAY because the applicable size can typically be derived from the context.

Please note further 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.



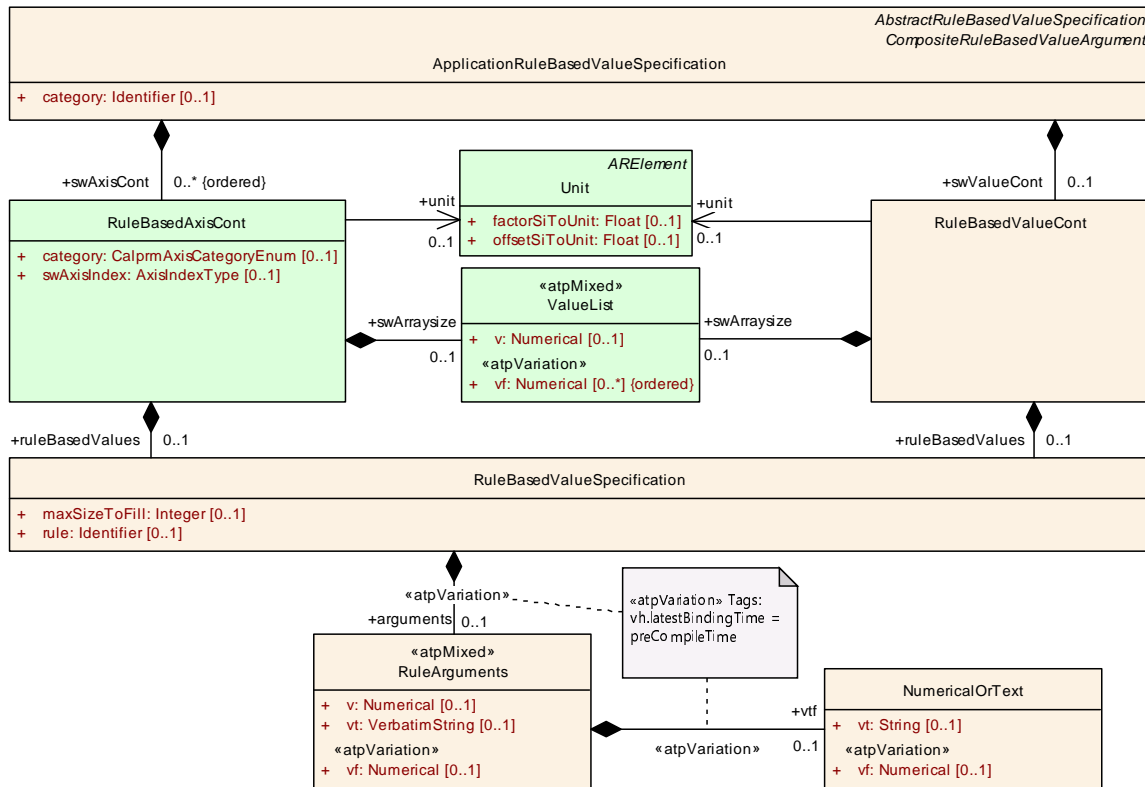


Figure 5.75: Definition of an **ApplicationRuleBasedValueSpecification**

[TPS\_SWCT\_02053] Values of **RuleBasedAxisCont** with the **category COM\_AXIS, RES\_AXIS** are for display only [In case of **ApplicationRuleBasedValueSpecification**s 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**.

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**.]()

Hence, values of the **COM\_AXIS** itself are described by **RuleBasedValueCont**.

[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].

<b>Class</b>	<b>NumericalRuleBasedValueSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	This meta-class is used to support a rule-based initialization approach for data types with an array-nature (ImplementationDataType of category ARRAY).			
<b>Base</b>	ARObject, <a href="#">AbstractRuleBasedValueSpecification</a> , <a href="#">ValueSpecification</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
ruleBasedValues	<a href="#">RuleBasedValueSpecification</a>	0..1	aggr	This represents the rule based value specification for the array. <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=false xml.typeWrapperElement=false

**Table 5.142: NumericalRuleBasedValueSpecification**

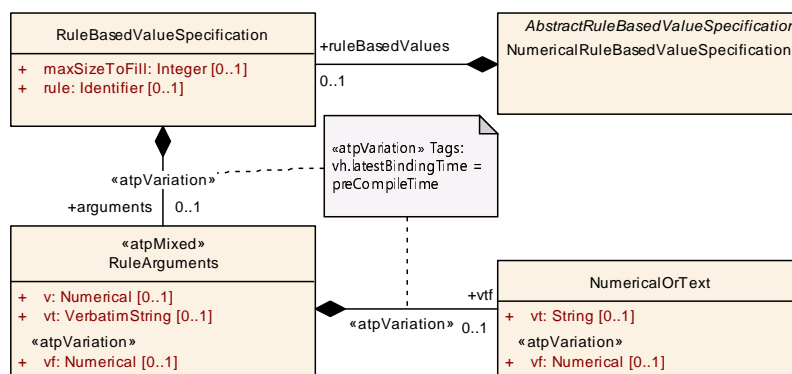
**[constr\_1925] Existence of [NumericalRuleBasedValueSpecification.ruleBasedValues](#)** [For each [NumericalRuleBasedValueSpecification](#), attribute [ruleBasedValues](#) shall exist **at the time when the contract phase generation is executed.**]()

**[TPS\_SWCT\_01495] Standardized value of [RuleBasedValueSpecification.rule](#)** [AUTOSAR reserves a dedicated value of [RuleBasedValueSpecification.rule](#) and [CompositeRuleBasedValueSpecification.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](#))



**Figure 5.76: Definition of an [NumericalRuleBasedValueSpecification](#)**

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 [RuleBasedValueSpecification.arguments](#) shall not exceed the array size [If the number of [RuleBasedValueSpecification.arguments](#) exceeds the number of elements of an array that it is applied to then the [RuleBasedValueSpecification.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 [RuleBasedValueSpecification.arguments](#) until the last element of the array [The following rule applies to [RuleBasedValueSpecifications](#) of rule **FILL\_UNTIL\_END**:

If the number of [RuleBasedValueSpecification.arguments](#) is smaller than the number of elements of the array it is applied to then the value of the last [RuleBasedValueSpecification.arguments](#) 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 [RuleBasedValueSpecification.arguments](#) until the number of elements specified in [maxSizeToFill](#) [The following rule applies to [RuleBasedValueSpecifications](#) of rule **FILL\_UNTIL\_MAX\_SIZE**:

If the number of [RuleBasedValueSpecification.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 [RuleBasedValueSpecification.arguments](#) shall be applied to so many of the following elements that the first [maxSizeToFill](#) elements of the array are filled.] ([RS\\_SWCT\\_03260](#))

<b>Class</b>	<b>RuleBasedValueSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	This meta-class is used to support a rule-based initialization approach for data types with an array-nature ( <a href="#">ApplicationArrayDataType</a> and <a href="#">ImplementationDataType</a> of category <b>ARRAY</b> ) or a compound <a href="#">ApplicationPrimitiveDataType</a> (which also boils down to an array-nature).			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	RuleBasedValueSpecification			
arguments	<a href="#">RuleArguments</a>	0..1	aggr	This represents the arguments for the RuleBasedValue Specification. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
maxSizeToFill	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. <b>Tags:</b> xml.sequenceOffset=40
rule	Identifier	0..1	attr	This denotes the name of the rule of the RuleBasedValue Specification. The rule determines the calculation specification according which the arguments are used to calculated the values. <b>Tags:</b> xml.sequenceOffset=20

**Table 5.143: RuleBasedValueSpecification**

**[constr\_1926] Existence of [RuleBasedValueSpecification.rule](#)** [For each [RuleBasedValueSpecification](#), attribute `rule` shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1927] Existence of [RuleBasedValueSpecification.arguments](#)** [For each [RuleBasedValueSpecification](#), the aggregation of [RuleArguments](#) in the role `arguments` shall exist **at the time when the contract phase generation is executed.**]()

Class	<<atpMixed>> <b>RuleArguments</b>			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This represents the arguments for a rule-based value specification.			
Base	<i>ARObject</i>			
Attribute	Type	Mult.	Kind	Note
v	Numerical	0..1	attr	This represents a numerical value for the RuleBased ValueSpecification.
vf	Numerical	0..1	attr	This represents a numerical value for the RuleBased ValueSpecification which may subject to variability. The latest binding time of the VariationPoint shall be pre CompileTime. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
vt	VerbatimString	0..1	attr	This represents a textual value for the RuleBasedValue Specification.
vtf	<a href="#">NumericalOrText</a>	0..1	aggr	This aggregation represents the ability to provide a value that is either numerical or text which existence is subject to variability. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 5.144: RuleArguments**

As an example for the application of a [RuleBasedValueSpecification](#), consider the following scenario:

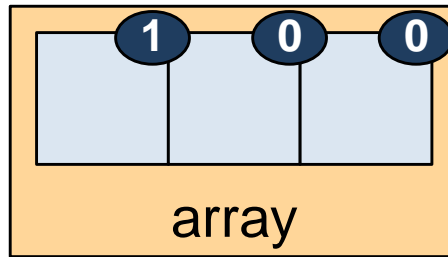


Figure 5.77: Value specification for a simple array

The sketched array depicted in Figure 5.77 corresponds to the modeling exemplified in Listing 5.23.

Listing 5.23: Value specification for a simple array

```

<ARRAY-VALUE-SPECIFICATION>
  <ELEMENTS>
    <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
      <SW-VALUE-CONT>
        <RULE-BASED-VALUES>
          <RULE>FILL_UNTIL_END</RULE>
          <ARGUMENTSS>
            <RULE-ARGUMENTS>
              <V>1</V>
              <V>0</V>
            </RULE-ARGUMENTS>
          </ARGUMENTSS>
        </RULE-BASED-VALUES>
      </SW-VALUE-CONT>
    </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
  </ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>
    
```

Please note that it is not foreseen that an `ArrayValueSpecification` consist of a collection of `ApplicationRuleBasedValueSpecification` or even a mixture of `ApplicationRuleBasedValueSpecification` with another kind of `ValueSpecification`.

**[constr\_10009] Aggregation of `ApplicationRuleBasedValueSpecification`**  
 [Each `ArrayValueSpecification` shall only aggregate at most one `ApplicationRuleBasedValueSpecification` in the role element.

If one `ApplicationRuleBasedValueSpecification` is aggregated then it shall be the only aggregated element, i.e. no further `ValueSpecification` shall exist in the same aggregation where an `ApplicationRuleBasedValueSpecification` is aggregated.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Please note that it is not foreseen that an `ArrayValueSpecification` consist of a collection of `NumericalRuleBasedValueSpecification` or even a mixture of

`NumericalRuleBasedValueSpecification` with another kind of `ValueSpecification`.

**[constr\_1754] Aggregation of `NumericalRuleBasedValueSpecification`**  
[Each `ArrayValueSpecification` shall only aggregate at most one `NumericalRuleBasedValueSpecification` in the role element.

If one `NumericalRuleBasedValueSpecification` is aggregated then it shall be the only aggregated element, i.e. no further `ValueSpecification` shall exist in the same aggregation where an `NumericalRuleBasedValueSpecification` is aggregated.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

### 5.6.6.2 Support for composite Data Types

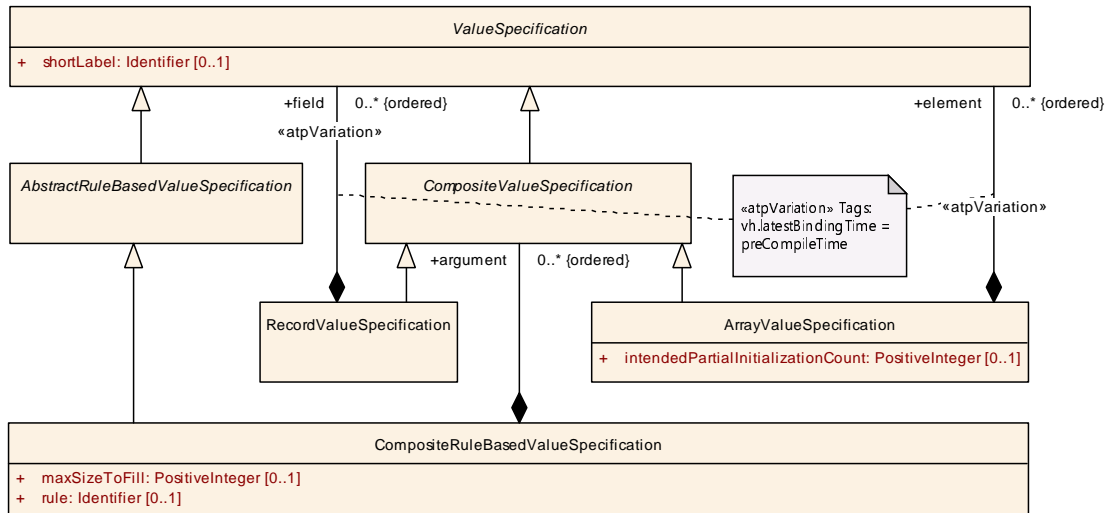
**[TPS\_SWCT\_01692] Meaning of `CompositeRuleBasedValueSpecification`**  
[The rule-based initialization of arrays of a composite data type is modeled by means of the `CompositeRuleBasedValueSpecification`.] (*RS\_SWCT\_03260*)

<b>Class</b>	<b>CompositeRuleBasedValueSpecification</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	This meta-class represents rule-based values for DataPrototypes typed by composite AutosarDataTypes.			
<b>Base</b>	<i>ARObject</i> , <i>AbstractRuleBasedValueSpecification</i> , <i>ValueSpecification</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
argument (ordered)	<code>CompositeValueSpecification</code>	*	aggr	This represents the collection of aggregated Value Specifications. The last ValueSpecification in the collection shall be taken to execute the filling rule. <b>Tags:</b> xml.sequenceOffset=30
compound Primitive Argument (ordered)	<code>CompositeRuleBasedValueArgument</code>	*	aggr	This represents the collection of aggregated Value Specifications for compound primitive data type. The last ValueSpecification in the collection shall be taken to execute the filling rule. <b>Tags:</b> xml.sequenceOffset=35
maxSizeToFill	PositiveInteger	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. <b>Tags:</b> xml.sequenceOffset=40
rule	Identifier	0..1	attr	This denotes the name of the rule of the RuleBasedValue Specification. The rule determines the calculation specification according which the arguments are used to calculated the values. <b>Tags:</b> xml.sequenceOffset=20

**Table 5.145: CompositeRuleBasedValueSpecification**

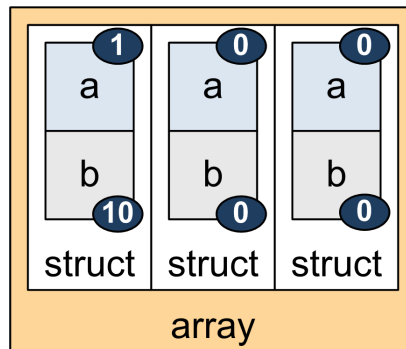
**[constr\_1928] Existence of `CompositeRuleBasedValueSpecification.rule`**  
[For each `CompositeRuleBasedValueSpecification`, attribute `rule` shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1929] Existence of CompositeRuleBasedValueSpecification.argument** [For each CompositeRuleBasedValueSpecification, the aggregation of CompositeValueSpecification in the role argument shall exist at the time when the contract phase generation is executed.]()



**Figure 5.78: Rule-based value specification of arrays of a composite data type**

As an example of how the rule-based initialization of composite data structures works, please consider the composite structure sketched in Figure 5.79. In simple terms, it describes an array consisting of elements that themselves are typed by a structure of two primitive elements.



**Figure 5.79: Example for the explanation of rule-based composite value-specification**

In this example, the element “a” of the first structure shall be initialized with the value 1, the corresponding “b” element shall be assigned a 10. All other values in all following elements shall be set to 0. This is also indicated by the numbers in ellipses in Figure 5.79.

The implementation of the example in ARXML is illustrated in Listing 5.24. As already explained before, the last (in the order of appearance) ValueSpecification in the context of an AbstractRuleBasedValueSpecification is taken to execute the rule (as described above).

**Listing 5.24: Example for composite rule-based value specification**

```

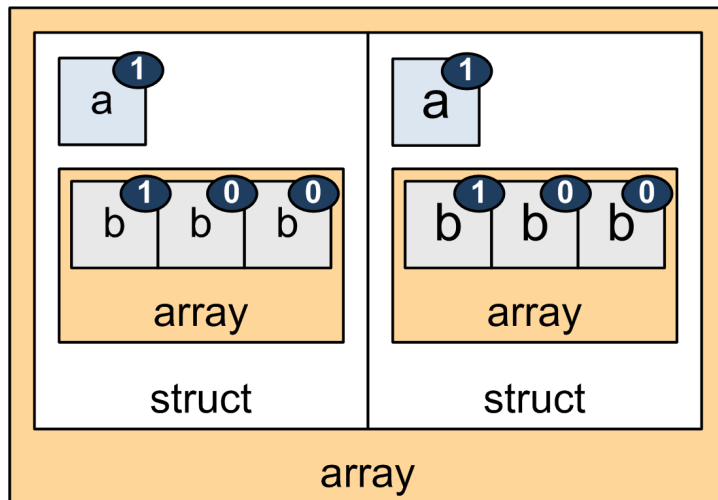
<ARRAY-VALUE-SPECIFICATION>
  <ELEMENTS>
    <COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
      <RULE>FILL_UNTIL_END</RULE>
      <ARGUMENTS>
        <RECORD-VALUE-SPECIFICATION>
          <FIELDS>
            <APPLICATION-VALUE-SPECIFICATION>
              <SW-VALUE-CONT>
                <SW-VALUES-PHYS>
                  <V>1</V>
                </SW-VALUES-PHYS>
              </SW-VALUE-CONT>
            </APPLICATION-VALUE-SPECIFICATION>
            <APPLICATION-VALUE-SPECIFICATION>
              <SW-VALUE-CONT>
                <SW-VALUES-PHYS>
                  <V>10</V>
                </SW-VALUES-PHYS>
              </SW-VALUE-CONT>
            </APPLICATION-VALUE-SPECIFICATION>
          </FIELDS>
        </RECORD-VALUE-SPECIFICATION>
        <RECORD-VALUE-SPECIFICATION>
          <FIELDS>
            <APPLICATION-VALUE-SPECIFICATION>
              <SW-VALUE-CONT>
                <SW-VALUES-PHYS>
                  <V>0</V>
                </SW-VALUES-PHYS>
              </SW-VALUE-CONT>
            </APPLICATION-VALUE-SPECIFICATION>
            <APPLICATION-VALUE-SPECIFICATION>
              <SW-VALUE-CONT>
                <SW-VALUES-PHYS>
                  <V>0</V>
                </SW-VALUES-PHYS>
              </SW-VALUE-CONT>
            </APPLICATION-VALUE-SPECIFICATION>
          </FIELDS>
        </RECORD-VALUE-SPECIFICATION>
      </ARGUMENTS>
    </COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
  </ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>

```

A more complicated example is sketched in Figure 5.80. Here, a deeply nested composite data structure is described: an array of structures that in turn contain an array.

To keep the ARXML listing as simple as possible, the example assumes that **all** (as opposed to the initialization of the first, and then of all other elements) “struct” elements shall be initialized with the same value.





**Figure 5.80: Value specification for a deeply nested array**

The deeply nested “array” is initialized by means of an [ApplicationRuleBased-ValueSpecification](#).

**Listing 5.25: Value specification for a deeply nested array**

```

<ARRAY-VALUE-SPECIFICATION>
  <ELEMENTS>
    <COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
      <RULE>FILL_UNTIL_END</RULE>
      <ARGUMENTS>
        <RECORD-VALUE-SPECIFICATION>
          <FIELDS>
            <APPLICATION-VALUE-SPECIFICATION>
              <SW-VALUE-CONT>
                <SW-VALUES-PHYS>
                  <V>1</V>
                </SW-VALUES-PHYS>
              </SW-VALUE-CONT>
            </APPLICATION-VALUE-SPECIFICATION>
          <ARRAY-VALUE-SPECIFICATION>
            <ELEMENTS>
              <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
                <SW-VALUE-CONT>
                  <RULE-BASED-VALUES>
                    <RULE>FILL_UNTIL_END</RULE>
                    <ARGUMENTSS>
                      <RULE-ARGUMENTS>
                        <V>1</V>
                        <V>0</V>
                      </RULE-ARGUMENTS>
                    </ARGUMENTSS>
                  </RULE-BASED-VALUES>
                </SW-VALUE-CONT>
              </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
            </ELEMENTS>
          </ARRAY-VALUE-SPECIFICATION>
        </RECORD-VALUE-SPECIFICATION>
      </ARGUMENTS>
    </COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
  </ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>

```

```

    </FIELDS>
  </RECORD-VALUE-SPECIFICATION>
</ARGUMENTS>
</COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
</ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>

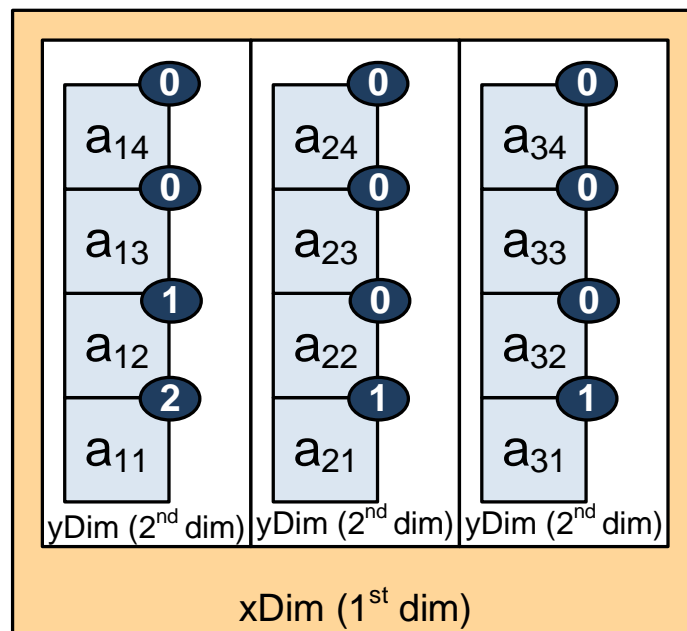
```

Please note that it is not foreseen that an `ArrayValueSpecification` consist of a collection of `CompositeRuleBasedValueSpecification` or even a mixture of `CompositeRuleBasedValueSpecification` with another kind of `ValueSpecification`.

**[constr\_1755] Aggregation of `CompositeRuleBasedValueSpecification`**  
 [Each `ArrayValueSpecification` shall only aggregate at most one `CompositeRuleBasedValueSpecification` in the role element.

If one `CompositeRuleBasedValueSpecification` is aggregated then it shall be the only aggregated element, i.e. no further `ValueSpecification` shall exist in the same aggregation where an `CompositeRuleBasedValueSpecification` is aggregated.

This rule shall be imposed **at the time when the contract phase generation is executed.** |()



**Figure 5.81: Rule-based Value specification for a 2-dimensional array**

As another variation of the theme, the rule-based initialization of multi-dimensional arrays shall be discussed.

Consequently, the following example extends the one-dimensional rule-based Value-Specification (see Figure 5.77) to a second dimension. The general layout is sketched in the following figure:

For the sake of clarity, the picture has been drawn to align the first dimension (the *x-axis*) of the two-dimensional array with the horizontal direction and the second dimension (the *y-axis*) with the vertical direction.

The direction index values of each array element are visible as subscript on the bottom right of the element, i.e.  $a_{12}$  indicates that the element is part of the first element on the *x-axis* and represents the second element of the *y-axis*. The initial value of element  $a_{12}$  shall be 1.

As indicated by the sketch in Figure 5.81, the second element (i.e. the “vertical” array, i.e. everything from  $a_{21}$  to  $a_{24}$ ) and all following (i.e. everything from  $a_{31}$  to  $a_{34}$ ) shall have the identical initial values. The first element deviates from the second in terms of initial values.

The creation of the `ArrayValueSpecification` for this example is based on the definition of a `CompositeRuleBasedValueSpecification` with two arguments:

- an `ArrayValueSpecification` that carries an `ApplicationRuleBasedValueSpecification` for the first element (that itself is an array) on the *x*-dimension and
- an `ArrayValueSpecification` that carries an `ApplicationRuleBasedValueSpecification` for each of the remaining elements (that itself are arrays) on the *x*-dimension.

#### Listing 5.26: Value specification for a 2-dimensional array

```

<ARRAY-VALUE-SPECIFICATION>
  <ELEMENTS>
    <COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
      <RULE>FILL_UNTIL_END</RULE>
      <ARGUMENTS>
        <ARRAY-VALUE-SPECIFICATION>
          <ELEMENTS>
            <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
              <CATEGORY>ARRAY</CATEGORY>
              <SW-VALUE-CONT>
                <RULE-BASED-VALUES>
                  <RULE>FILL_UNTIL_END</RULE>
                  <ARGUMENTSS>
                    <RULE-ARGUMENTS>
                      <V>2</V>
                      <V>1</V>
                      <V>0</V>
                    </RULE-ARGUMENTS>
                  </ARGUMENTSS>
                </SW-VALUE-CONT>
              </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
            </ELEMENTS>
          </ARRAY-VALUE-SPECIFICATION>
        <ARRAY-VALUE-SPECIFICATION>
          <ELEMENTS>
            <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
              <CATEGORY>ARRAY</CATEGORY>

```

```

<SW-VALUE-CONT>
  <RULE-BASED-VALUES>
    <RULE>FILL_UNITL_END</RULE>
    <ARGUMENTSS>
      <RULE-ARGUMENTS>
        <V>1</V>
        <V>0</V>
      </RULE-ARGUMENTS>
    </ARGUMENTSS>
  </RULE-BASED-VALUES>
</SW-VALUE-CONT>
</APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
</ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>
</ARGUMENTS>
</COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
</ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>
    
```

The next example adds one dimension to the array structure, i.e. it describes a three-dimensional array, as sketched in Figure 5.82.

In this case the *x-axis* has again been aligned in the horizontal direction, why the *y-axis* is drawn vertically. The *z-axis*, finally, goes horizontal again.

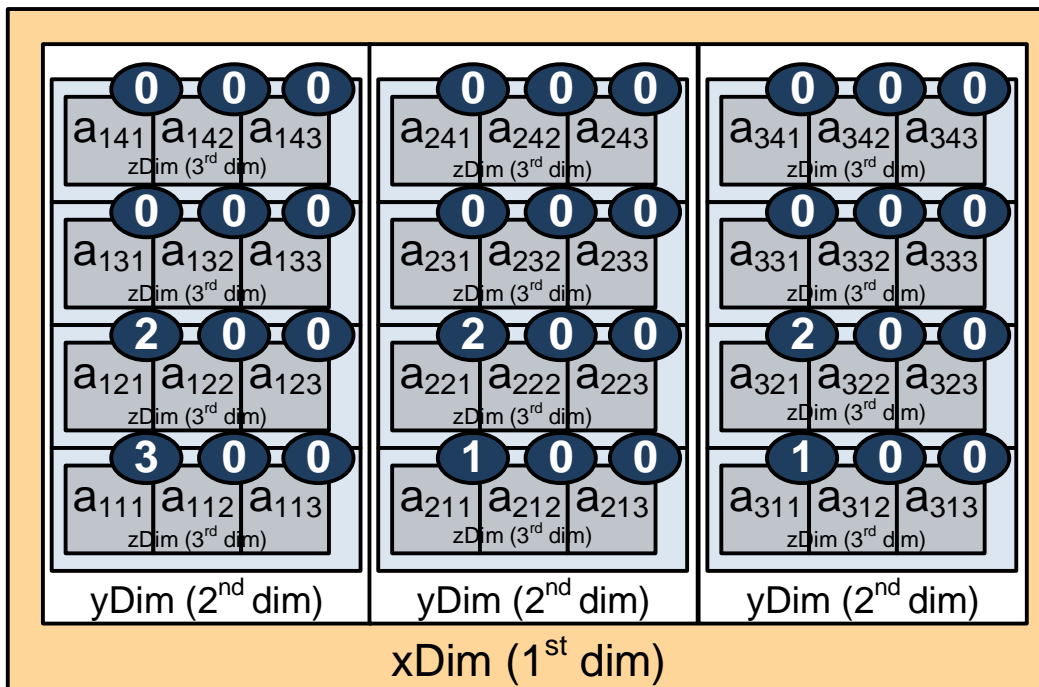


Figure 5.82: Rule-based Value specification for a 3-dimensional array

The principal structure of this example is similar to the previous example.

The first array element in the *x-dimension* (which includes everything from  $a_{111}$  to  $a_{143}$ ) defines a different initial value than the other elements.

Please note that the initial values of the second and third element in x-direction are identical.

**Listing 5.27: Value specification for a 3-dimensional array**

```

<ARRAY-VALUE-SPECIFICATION>
  <ELEMENTS>
    <COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
      <RULE>FILL_UNTIL_END</RULE>
      <ARGUMENTS>
        <ARRAY-VALUE-SPECIFICATION>
          <ELEMENTS>
            <COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
              <RULE>FILL_UNTIL_END</RULE>
              <ARGUMENTS>
                <ARRAY-VALUE-SPECIFICATION>
                  <ELEMENTS>
                    <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
                      <CATEGORY>ARRAY</CATEGORY>
                      <SW-VALUE-CONT>
                        <RULE-BASED-VALUES>
                          <RULE>FILL_UNTIL_END</RULE>
                          <ARGUMENTSS>
                            <RULE-ARGUMENTS>
                              <V>3</V>
                              <V>0</V>
                            </RULE-ARGUMENTS>
                          </ARGUMENTSS>
                        </RULE-BASED-VALUES>
                      </SW-VALUE-CONT>
                    </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
                  </ELEMENTS>
                </ARRAY-VALUE-SPECIFICATION>
              <ARRAY-VALUE-SPECIFICATION>
                <ELEMENTS>
                  <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
                    <CATEGORY>ARRAY</CATEGORY>
                    <SW-VALUE-CONT>
                      <RULE-BASED-VALUES>
                        <RULE>FILL_UNTIL_END</RULE>
                        <ARGUMENTSS>
                          <RULE-ARGUMENTS>
                            <V>2</V>
                            <V>0</V>
                          </RULE-ARGUMENTS>
                        </ARGUMENTSS>
                      </RULE-BASED-VALUES>
                    </SW-VALUE-CONT>
                  </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
                </ELEMENTS>
              </ARRAY-VALUE-SPECIFICATION>
            <ARRAY-VALUE-SPECIFICATION>
              <ELEMENTS>
                <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
                  <CATEGORY>ARRAY</CATEGORY>
                  <SW-VALUE-CONT>

```

```

<RULE-BASED-VALUES>
  <RULE>FILL_UNTIL_END</RULE>
  <ARGUMENTSS>
    <RULE-ARGUMENTS>
      <V>0</V>
    </RULE-ARGUMENTS>
  </ARGUMENTSS>
</RULE-BASED-VALUES>
</SW-VALUE-CONT>
</APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
</ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>
</ARGUMENTS>
</COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
</ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>
<ARRAY-VALUE-SPECIFICATION>
  <ELEMENTS>
    <COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
      <RULE>FILL_UNTIL_END</RULE>
      <ARGUMENTS>
        <ARRAY-VALUE-SPECIFICATION>
          <ELEMENTS>
            <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
              <CATEGORY>ARRAY</CATEGORY>
              <SW-VALUE-CONT>
                <RULE-BASED-VALUES>
                  <RULE>FILL_UNTIL_END</RULE>
                  <ARGUMENTSS>
                    <RULE-ARGUMENTS>
                      <V>1</V>
                      <V>0</V>
                    </RULE-ARGUMENTS>
                  </ARGUMENTSS>
                </RULE-BASED-VALUES>
              </SW-VALUE-CONT>
            </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
          </ELEMENTS>
        </ARRAY-VALUE-SPECIFICATION>
      </ARRAY-VALUE-SPECIFICATION>
    </ELEMENTS>
    <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
      <CATEGORY>ARRAY</CATEGORY>
      <SW-VALUE-CONT>
        <RULE-BASED-VALUES>
          <RULE>FILL_UNTIL_END</RULE>
          <ARGUMENTSS>
            <RULE-ARGUMENTS>
              <V>2</V>
              <V>0</V>
            </RULE-ARGUMENTS>
          </ARGUMENTSS>
        </RULE-BASED-VALUES>
      </SW-VALUE-CONT>
    </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
  </ELEMENTS>

```

```

</ARRAY-VALUE-SPECIFICATION>
<ARRAY-VALUE-SPECIFICATION>
  <ELEMENTS>
    <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
      <CATEGORY>ARRAY</CATEGORY>
      <SW-VALUE-CONT>
        <RULE-BASED-VALUES>
          <RULE>FILL_UNTIL_END</RULE>
          <ARGUMENTSS>
            <RULE-ARGUMENTS>
              <V>0</V>
            </RULE-ARGUMENTS>
          </ARGUMENTSS>
        </RULE-BASED-VALUES>
      </SW-VALUE-CONT>
    </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
  </ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>
</ARGUMENTS>
</COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
</ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>
</ARGUMENTS>
</COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
</ELEMENTS>
</ARRAY-VALUE-SPECIFICATION>

```

### 5.6.6.3 Support for compound primitive Data Types

Please note that the `CompositeRuleBasedValueSpecification` also supports the rule-based initialization of arrays<sup>20</sup> that are typed by a `Compound Primitive Data Type`, e.g. of category `CURVE`.

#### [TPS\_SWCT\_01836] Attributes of `CompositeRuleBasedValueSpecification`

[Meta-class `CompositeRuleBasedValueSpecification` can be used to fulfill two distinct use cases:

- Definition of a rule-based value specification for an array of composite data type based on the aggregation of `CompositeValueSpecification` in the role `argument`.
- Definition of a rule-based value specification for an array of `Compound Primitive Data Type` based on the aggregation of `CompositeRuleBasedValueArgument` in the role `compoundPrimitiveArgument`.

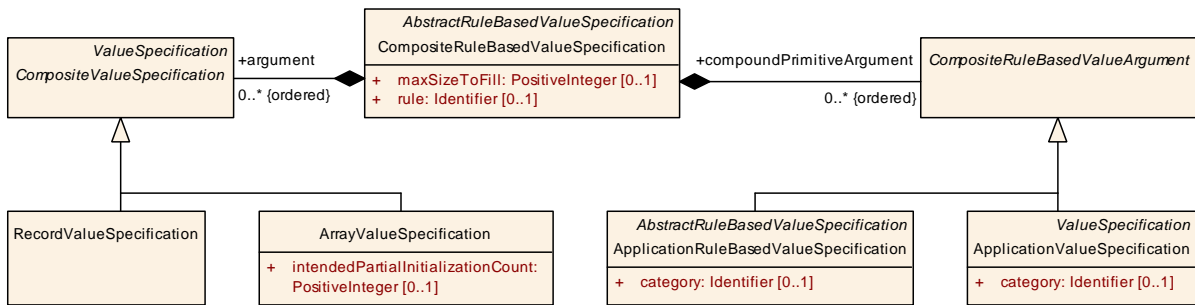
]0

<sup>20</sup>Of course, this capability is restricted to `ApplicationArrayDataType`. On the level of `ImplementationDataType`, `ValueSpecifications` that reflect the structure of the respective `ImplementationDataType` are used.

[constr\_10075] Existence of **CompositeRuleBasedValueSpecification.argument** vs. **compoundPrimitiveArgument** [For every **CompositeRuleBasedValueSpecification**, at most one of the aggregations

- `argument`
- `compoundPrimitiveArgument`

shall exist at the time when the contract phase generation is executed.]()



**Figure 5.83: Rule-based value specification for compound primitive data objects**

<b>Class</b>	<i>CompositeRuleBasedValueArgument</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	This meta-class has the ability to serve as the abstract base class for ValueSpecifications that can be used for compound primitive data types.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	ApplicationRuleBasedValueSpecification, ApplicationValueSpecification			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 5.146: CompositeRuleBasedValueArgument**

Please find a simplified example for the rule-based initialization of an array of an `ApplicationPrimitiveDataType` of category `CURVE` with an integrated axis in section 5.6.7.5.

## 5.6.7 Examples

### 5.6.7.1 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 `swArraysizes` 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 5.28: 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>
      <VF>2</VF>
      <VF>3</VF>
      </SW-VALUES-PHYS>
    </SW-AXIS-CONT>
    </SW-AXIS-CONTS>
    <SW-VALUE-CONT>
    <UNIT-REF DEST="UNIT">/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>
    
```

### 5.6.7.2 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 5.29: 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"/>/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>

```

### 5.6.7.3 Example for Constant Specification for MAP with two STD\_AXIS

The example contained in this sub-chapter illustrates the creation of the `ConstantSpecification` for a MAP that (in contrast to the previous example sketched in Listing 5.29) consists of two `STD_AXIS`.

Like in the previous example, the `v` attribute is used for the `swArraysize` as well as for the `swValuesPhys`.

**Listing 5.30: Example for Constant Specification for STD\_AXIS**

```
<CONSTANT-SPECIFICATION>
  <SHORT-NAME>MapExample</SHORT-NAME>
  <VALUE-SPEC>
    <APPLICATION-VALUE-SPECIFICATION>
      <CATEGORY>MAP</CATEGORY>
      <SW-AXIS-CONTS>
        <SW-AXIS-CONT>
          <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
          <SW-ARRAYSIZE>
            <V>4</V>
          </SW-ARRAYSIZE>
          <SW-VALUES-PHYS>
            <V>1</V>
            <V>2</V>
            <V>3</V>
            <V>4</V>
          </SW-VALUES-PHYS>
        </SW-AXIS-CONT>
        <SW-AXIS-CONT>
          <SW-AXIS-INDEX>2</SW-AXIS-INDEX>
          <SW-ARRAYSIZE>
            <V>2</V>
          </SW-ARRAYSIZE>
          <SW-VALUES-PHYS>
            <V>10</V>
            <V>11</V>
          </SW-VALUES-PHYS>
        </SW-AXIS-CONT>
      </SW-AXIS-CONTS>
      <SW-VALUE-CONT>
        <SW-ARRAYSIZE>
          <V>4</V>
          <V>2</V>
        </SW-ARRAYSIZE>
        <SW-VALUES-PHYS>
          <VG>
            <LABEL>
              <L-4 L="EN">Values for 10</L-4>
            </LABEL>
            <V>110</V>
            <V>210</V>
            <V>310</V>
            <V>410</V>
          </VG>
        </SW-VALUES-PHYS>
      </SW-VALUE-CONT>
    </APPLICATION-VALUE-SPECIFICATION>
  </VALUE-SPEC>
</CONSTANT-SPECIFICATION>
```

```

<LABEL>
  <L-4 L="EN">Values for 11</L-4>
</LABEL>
<V>111</V>
<V>211</V>
<V>311</V>
<V>411</V>
</VG>
</SW-VALUES-PHYS>
</SW-VALUE-CONT>
</APPLICATION-VALUE-SPECIFICATION>
</VALUE-SPEC>
</CONSTANT-SPECIFICATION>

```

#### 5.6.7.4 Example for Constant Specification for COM\_AXIS

The following example illustrates how an [ConstantSpecification](#) is specified for a [COM\\_AXIS](#).

**Listing 5.31: 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">/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>

```

### 5.6.7.5 Example for Constant Specification for an Array of compound primitive Objects

The example starts with the definition of the data type for the input value for the curve see Listing 5.32. This data type is used for the definition of the actual curve data type in Listing 5.34.

**Listing 5.32: Definition of curve input value data type**

```
<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>axisInputType</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-DATA-TYPE>
```

The next step is the definition of the data type for the result of the interpolation. This part is sketched in Listing 5.33. This data type is used for the definition of the actual curve data type in Listing 5.34.

**Listing 5.33: Definition of data type for the result of the interpolation**

```
<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>curveType</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-DATA-TYPE>
```

Finally, the data type for the actual curve is defined, see Listing 5.34.

**Listing 5.34: Definition of curve data type**

```
<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>MyTable</SHORT-NAME>
  <CATEGORY>CURVE</CATEGORY>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
      <SW-CALPRM-AXIS-SET>
        <SW-CALPRM-AXIS>
          <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
          <CATEGORY>STD_AXIS</CATEGORY>
        <SW-AXIS-INDIVIDUAL>
```

```

        <INPUT-VARIABLE-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-
            TYPE"/>/ApplicationDataTypes/axisInputType</INPUT-VARIABLE-
            TYPE-REF>
        <SW-MAX-AXIS-POINTS>10</SW-MAX-AXIS-POINTS>
        <SW-MIN-AXIS-POINTS>0</SW-MIN-AXIS-POINTS>
    </SW-AXIS-INDIVIDUAL>
    <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
</SW-CALPRM-AXIS>
</SW-CALPRM-AXIS-SET>
<VALUE-AXIS-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"/>/
    ApplicationDataTypes/curveType</VALUE-AXIS-DATA-TYPE-REF>
</SW-DATA-DEF-PROPS-CONDITIONAL>
</SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-DATA-TYPE>
    
```

The `ApplicationPrimitiveDataType` of category `CURVE` defined in the listing above may be used as the element type (`ApplicationArrayElement.type`) of an `ApplicationArrayDataType`. The definition of this array data type is not part of the example.

The initialization of a `DataPrototype` typed by such an `ApplicationArray-DataType` is sketched in the listing below:

#### Listing 5.35: Definition of constant to initialize an array of curve data type

```

<CONSTANT-SPECIFICATION>
    <SHORT-NAME>MyConst</SHORT-NAME>
    <VALUE-SPEC>
        <ARRAY-VALUE-SPECIFICATION>
            <ELEMENTS>
                <COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
                    <RULE>FILL_UNTIL_END</RULE>
                    <COMPOUND-PRIMITIVE-ARGUMENTS>
                        <APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
                            <SW-AXIS-CONTS>
                                <RULE-BASED-AXIS-CONT>
                                    <SW-ARRAYSIZE>
                                        <V>10</V>
                                    </SW-ARRAYSIZE>
                                    <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
                                <RULE-BASED-VALUES>
                                    <ARGUMENTSS>
                                        <RULE-ARGUMENTS>
                                            <V>1</V>
                                            <V>2</V>
                                            <V>3</V>
                                            <V>4</V>
                                            <V>5</V>
                                            <V>6</V>
                                            <V>7</V>
                                            <V>8</V>
                                            <V>9</V>
                                            <V>10</V>
                                        </RULE-ARGUMENTS>
                                    </ARGUMENTSS>
                                </SW-AXIS-CONTS>
                            </APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
                        </COMPOUND-PRIMITIVE-ARGUMENTS>
                    </COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
                </ELEMENTS>
            </ARRAY-VALUE-SPECIFICATION>
        </VALUE-SPEC>
    </CONSTANT-SPECIFICATION>
    
```

```

    </RULE-BASED-VALUES>
  </RULE-BASED-AXIS-CONT>
</SW-AXIS-CONTS>
<SW-VALUE-CONT>
  <SW-ARRAYSIZE>
    <V>10</V>
  </SW-ARRAYSIZE>
<RULE-BASED-VALUES>
  <RULE>FILL_UNTIL_END</RULE>
  <ARGUMENTSS>
    <RULE-ARGUMENTS>
      <V>0</V>
    </RULE-ARGUMENTS>
  </ARGUMENTSS>
</RULE-BASED-VALUES>
</SW-VALUE-CONT>
</APPLICATION-RULE-BASED-VALUE-SPECIFICATION>
</COMPOUND-PRIMITIVE-ARGUMENTS>
</COMPOSITE-RULE-BASED-VALUE-SPECIFICATION>
</ELEMENTS>
</ARRAY-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 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!]()

**[constr\_1706] Definition of initial value for data transmission** [Initial values for data transmission shall only be defined by means of `NonqueuedSenderComSpec.initValue` resp. `NonqueuedReceiverComSpec.initValue`. Any definition of an `initValue` defined in the context of `VariableDataPrototype` shall be ignored.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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`. A restriction applies, please consult with [\[constr\\_1706\]](#).
2. It is possible to aggregate an `initValue` at the level of a `ComSpec`, namely:
  - `NonqueuedSenderComSpec`
  - `NonqueuedReceiverComSpec`
  - `ParameterProvideComSpec`
  - `ParameterRequireComSpec`
  - `NvRequireComSpec`
  - `NvProvideComSpec`
3. It is possible to aggregate a `implInitValue` and an `applInitValue` 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 3 supersedes a definition on level 2.]()

## 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.swValuesPhys.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 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](#))

Please note that the definition of Compound Primitive Data Type can be found in section 5.6.

**[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`, `shortLabel`, `symbol`) defined by the applicable `CompuScales`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Please note that several attributes of meta-class `CompuScale` can be taken to describe the actual value. It is therefore necessary to clarify what happens if several of these attributes exist within the context of one `CompuScale`. This clarification can be found in [TPS\_SWCT\_01696].

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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` (that may contain an `ApplicationRuleBasedValueSpecification`).

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

### 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`.]()

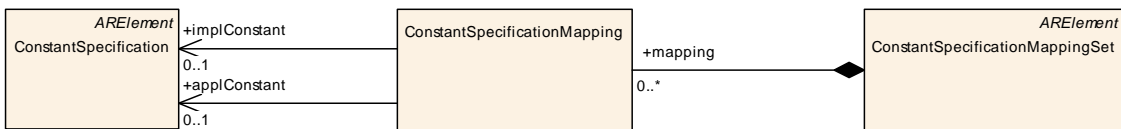
**[constr\_1029] ConstantSpecificationMapping and ConstantSpecification** [It is required that one `ConstantSpecification` referenced from a `ConstantSpecificationMapping` needs to be defined in the application domain (`applConstant`) and the other referenced `ConstantSpecification` needs to be defined in the implementation domain (`implConstant`).

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

[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**:

- [TPS\_SWCT\_01840] A **ParameterSwComponentType** references a **ConstantSpecificationMappingSet** [ParameterSwComponentType: here, the **ConstantSpecificationMappingSet** is directly associated by the **ParameterSwComponentType**.]()
- [TPS\_SWCT\_01841] A **NvBlockSwComponentType** references a **ConstantSpecificationMappingSet** [NvBlockSwComponentType: in this case, the **ConstantSpecificationMappingSet** is associated with the aggregated **NvBlockDescriptor**.]()



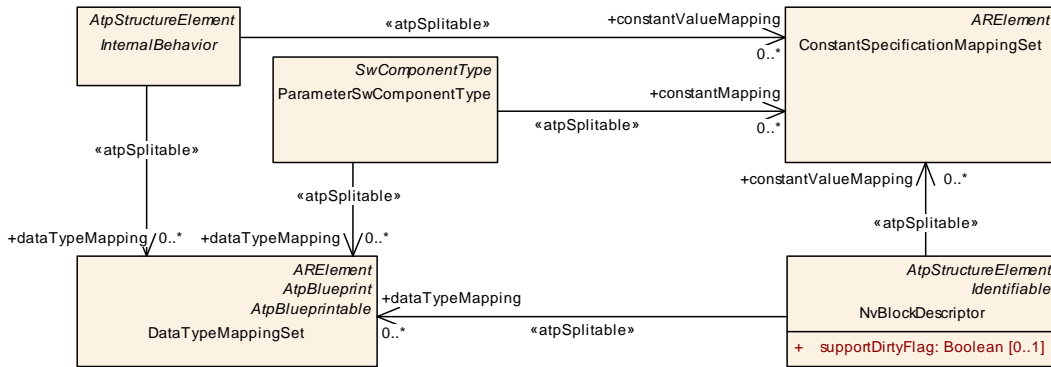
**Figure 5.84: Constant Mapping**

<b>Class</b>	<b>ConstantSpecificationMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	This meta-class is used to create an association of two ConstantSpecifications. One Constant Specification 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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
applConstant	ConstantSpecification	0..1	ref	A ConstantSpecification defined in the application domain.
implConstant	ConstantSpecification	0..1	ref	A ConstantSpecification defined in the implementation domain.

**Table 5.147: ConstantSpecificationMapping**

[constr\_1931] Existence of **ConstantSpecificationMapping.applConstant** [For each **ConstantSpecificationMapping**, the reference to meta-class **ConstantSpecification** in the role **applConstant** shall exist **at the time when the contract phase generation is executed**.]()

[constr\_1932] Existence of **ConstantSpecificationMapping.implConstant** [For each **ConstantSpecificationMapping**, the reference to meta-class **ConstantSpecification** in the role **implConstant** shall exist **at the time when the contract phase generation is executed**.]()



**Figure 5.85: Relation between data type mapping and constant mapping**

Class	<b>ConstantSpecificationMappingSet</b>			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents the ability to map two ConstantSpecifications to each others. One Constant Specification is supposed to be described in the application domain and the other should be described in the implementation domain. <b>Tags:</b> atp.recommendedPackage=ConstantSpecificationMappingSets			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
mapping	<a href="#">ConstantSpecificationMapping</a>	*	aggr	ConstantSpecificationMappings owned by the ConstantSpecificationMappingSet.

**Table 5.148: 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:

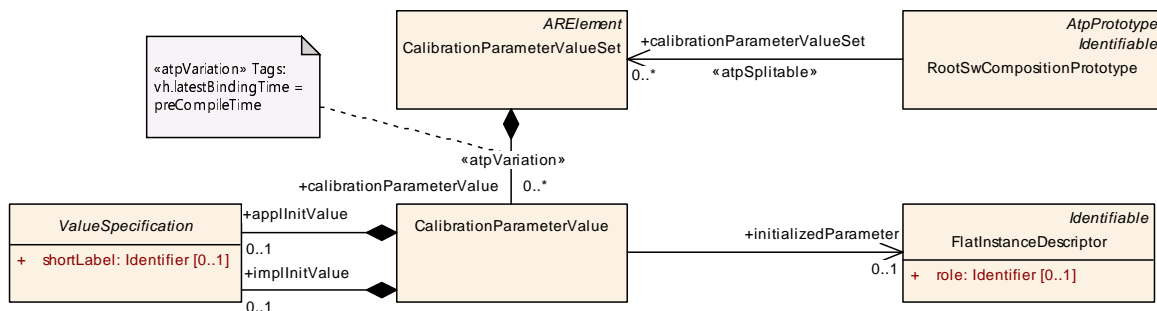
- [applInitValue](#) for data structured according to [ApplicationDataType](#). In this case the values are defined in the physical domain.
- [implInitValue](#) for data structured according to [ImplementationDataType](#). In this case the values are defined in the numerical domain.

]([RS\\_SWCT\\_03175](#))

Anyhow, these initial values can be imported from e.g. an ASAM CDF file.

<b>Class</b>	<b>CalibrationParameterValueSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::CalibrationParameter Values			
<b>Note</b>	Specification of a constant that can be part of a package, i.e. it can be defined stand-alone. <b>Tags:</b> atp.recommendedPackage=CalibrationParameterValueSets			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
calibration ParameterValue	CalibrationParameter Value	*	aggr	This represents single CalibrationParameterValues in the CalibrationParameterValueSet. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 5.149: CalibrationParameterValueSet**



**Figure 5.86: Calibration Parameter Values**

<b>Class</b>	<b>CalibrationParameterValue</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::CalibrationParameter Values			
<b>Note</b>	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 an 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.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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
initialized Parameter	FlatInstanceDescriptor	0..1	ref	This represents the parameter that is initialized by the CalibrationParameterValue.

**Table 5.150: CalibrationParameterValue**

[constr\_1933] Existence of **CalibrationParameterValue.initializedParameter** [For each **CalibrationParameterValue**, the reference to meta-class

`ConstantSpecification` in the role `initializedParameter` shall exist **at the time when the contract phase generation is executed.**]()

## 5.7.5 Initial Value for optional Element

The existence of optional elements in a given `AutosarDataType` needs to be properly considered for the definition of an initial value.

### 5.7.5.1 Initial Value for optional `ApplicationRecordElement`

The initial value for a given `DataPrototype` typed by an `ApplicationRecordDataType` is defined by means of a `RecordValueSpecification`.

**[TPS\_SWCT\_01823] Definition of `ValueSpecification` for an `ApplicationRecordDataType` with unavailable optional elements** [If an `ApplicationRecordDataType` contains `ApplicationRecordElements` where attribute `isOptional` is set to `True`, it is still necessary the corresponding `RecordValueSpecifications` fulfill [\[constr\\_1271\]](#), i.e. a `ValueSpecification` shall be provided for optional elements that are unavailable as far as the `ValueSpecification` is concerned.

The canonical approach to fulfill [\[constr\\_1271\]](#) is to fill the place in the structure that represents a non-available optional element of the structure with a special kind of `ValueSpecification` (the `NotAvailableValueSpecification`) in order to unambiguously convey the information that this element is not available for the specific `DataPrototype`, as far as the initial value is concerned.]([RS\\_SWCT\\_03320](#))

**[TPS\_SWCT\_01821] Semantics of attribute `NotAvailableValueSpecification.defaultPattern`** [The usage of `NotAvailableValueSpecification` explicitly defines that no specific initialization value shall be defined for an optional element.

The memory area occupied by the `NotAvailableValueSpecification` shall however be filled with a deterministic pattern during initialization.

The content of `defaultPattern` shall be used to fill the gap in the memory occupied by an optional element with `NotAvailableValueSpecification` initialization.]([RS\\_SWCT\\_03320](#))

**[TPS\_SWCT\_01822] Application of attribute `NotAvailableValueSpecification.defaultPattern` happens only during initialization** [The value of `NotAvailableValueSpecification.defaultPattern` is applied only during the initialization of the respective `DataPrototype`.

Therefore, if the optional element is not received during a specific reception, the memory area is untouched. It is the duty of the application to check whether that optional element has actually been received.](RS\_SWCT\_03320)

**[constr\_10005] Existence of attribute `NotAvailableValueSpecification.defaultPattern`** [For each `NotAvailableValueSpecification`, attribute `defaultPattern` shall exist **at the time when the contract phase generation is executed.**]([RS\\_SWCT\\_03320](#))

**[constr\_10006] Valid interval of attribute `NotAvailableValueSpecification.defaultPattern`** [The valid interval for attribute `NotAvailableValueSpecification.defaultPattern` **at the time when the contract phase generation is executed** is 0..255.]([RS\\_SWCT\\_03320](#))

### 5.7.5.2 Initial Value for optional `ImplementationDataTypeElement`

**[TPS\_SWCT\_01785] Initial value for `ImplementationDataType` of category `STRUCTURE` where attribute `isStructWithOptionalElement` set to the value `True`** [If an initial value is to be provided for an `ImplementationDataType` of category `STRUCTURE` where attribute `isStructWithOptionalElement` set to the value `True` then an initial value shall be defined for all `ImplementationDataTypeElements` including the first `ImplementationDataTypeElement` where the `shortName` is set to the value `availabilityBitfield`.]([RS\\_SWCT\\_03320](#))

**[TPS\_SWCT\_01786] Initial value for the `ImplementationDataTypeElement` where the `shortName` is set to the value `availabilityBitfield`** [The initial value for the `ImplementationDataTypeElement` where the `shortName` is set to the value `availabilityBitfield` shall be defined in a way that the bit that represents the existence of a given element is set to `True` if the element shall initially be available.

If the corresponding element shall not be initially available then the respective bit shall be set to `False`.]([RS\\_SWCT\\_03320](#))

**[TPS\_SWCT\_01787] Initialization of not-available `ImplementationDataTypeElement`** [If a given `ImplementationDataTypeElement` is not available in the context of the definition of an initial value then a “dummy” initial value shall be defined anyway for the element in order to not break [[constr\\_1272](#)].

The provided `ValueSpecification` shall be considered as “don’t care”.]([RS\\_SWCT\\_03320](#))

From the perspective of performance, it is recommended to use the value 0 for an initialization according to [[TPS\\_SWCT\\_01787](#)] of non-available `ImplementationDataTypeElement`.

## 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 `AutosarDataTypes` are set up, then the rules for the different types of `PortInterfaces` are derived.

Another aspect of compatibility is 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 **at the time when the RTE is generated** if and only if one of the following conditions applies:

1. All the following sub conditions apply:



- (a) They have the same `category`.
  - (b) The `swDataDefProps` attached to the M1 data types are compatible.
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 the meaning of “`swDataDefProps` attached to the M1 data types are compatible” is explained in section 6.2.4.

Please note further 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 **at the time when the RTE is generated** 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. For each `ApplicationRecordDataType.element`, the attribute `isOptional` shall either
  - not exist on both sides or
  - be set to the value `False` if it only exists on one side or
  - have the identical value on both sides.

3. 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 **at the time when the RTE is generated** if and only if one of the following conditions applies:

1. All the following sub conditions 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 **at the time when the RTE is generated** if and only if after all type-references are resolved one of the following rules apply:

1. All the following sub conditions apply:
  - (a) They have the same `category`.
  - (b) They have the identical structure (this refers to `ImplementationDataTypeElement` and their `subElements`).

- (c) The attributes `arraySize` and `arraySizeSemantics` have (given the existence) identical values.
  - (d) For each `ImplementationDataType.subElement`, the attribute `isOptional` shall either
    - not exist on both sides or
    - be set to the value `False` if it only exists on one side or
    - have the identical value on both sides.
  - (e) The `swDataDefProps` attached to the M1 data types are compatible.
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 the meaning of “`swDataDefProps` attached to the M1 data types are compatible” is explained in section 6.2.4.

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 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]).

This rule shall be imposed **at the time when the RTE is generated.**]()

### 6.2.3 Compatibility of SwBaseType

**[constr\_1220] Compatibility of SwBaseType** [Two `SwBaseType`s are compatible **at the time when the RTE is generated** if and only if attributes

- `baseTypeSize` respectively
- `byteOrder`,
- `memAlignment`,
- `baseTypeEncoding`, and
- `nativeDeclaration`

have identical values **at the time when the RTE is generated.**]()

### 6.2.4 Compatibility of SwDataDefProps

**[constr\_1051] Compatibility of SwDataDefProps** [`SwDataDefProps` are compatible **at the time when the RTE is generated** 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 or neither of them associates such a method.
3. They both aggregate a `ValueSpecification` in the role `invalidValue` or neither of them aggregates a `ValueSpecification` in the role `invalidValue`.
4. If existent (see previous condition), 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 effectively identical, e.g. if one `ArrayValueSpecification` specifies all values explicitly and the other `ArrayValueSpecification` specifies values based on a rule then the yield of both `ArrayValueSpecifications` (i.e. element for element) shall be 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] applies<sup>1</sup>.

5. They refer to compatible data constraints `dataConstr`.

6. They refer to compatible `swRecordLayouts`

All other attributes (e.g. `swCalibrationAccess` do not affect compatibility).]()

Please note that compatible conversion methods are described in chapter 6.2.4.5.

### 6.2.4.1 Compatibility of Units

**[constr\_1052] Compatibility of Units** [Two `Unit` definitions are compatible **at the time when the RTE is generated** 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`:

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed**.]()

---

<sup>1</sup>if 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\_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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1393] Existence of RuleBasedValueCont.unit** [For every `RuleBasedValueCont`, the reference `unit` shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1771] Existence of SwValueCont.unit** [For every `SwValueCont`, the reference `unit` shall exist **at the time when the contract phase generation is executed.**]()

#### 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`.

The imposition time of this constraint depends on the context:

- If the compatibility of `PhysicalDimensions` is evaluated in the context of the creation of a `SwConnector`, then the rule shall be imposed **at the time when the RTE is generated.**
- If the context is the creation of an `ApplicationValueSpecification`, then the rule shall be imposed **at the time when the contract phase generation is executed.**

]()

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.

For client-server communication, the following rules apply:

- For `arguments` with attribute `direction` set to the value `in`, the client shall take the role of the *provider* and the server shall take the role of the *requiring side*.
- For `arguments` with attribute `direction` set to the value `inout` the `DataConstr` shall be equal on both sides.
- For `arguments` with attribute `direction` set to the value `out`, the server shall take the role of the *provider* and the client shall take the role of the *requiring side*.

This rule shall be applied **at the time when the RTE is generated.**]()

In addition, it is always allowed that 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 **at the time when the RTE is generated.**]()

**[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 **at the time when the RTE is generated.**]()

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 VALUE`: **[constr\_1055] ImplementationDataType has category VALUE** [The attributes `baseType` shall refer to a compatible `SwBaseType` **at the time when the contract phase generation is executed.**]() (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 **at the time when the contract phase generation is executed.**]()
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 **at the time when the contract phase generation is executed.**]() (in other words, the target types of the pointers shall be identical, not only compatible).
4. `category FUNCTION_REFERENCE`: **[constr\_1058] ImplementationDataType has category FUNCTION\_REFERENCE** [The attributes `SwDataDefProps.swPointerTargetProps.functionPointerSignature` shall refer to `BswModuleEntry`s which each resolve to the **same function signature** **at the time when the contract phase generation is executed.**]()

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 **at the time when the RTE is generated** 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:

- `TEXTTABLE`
- `SCALE_LINEAR_AND_TEXTTABLE`
- `SCALE_RATIONAL_AND_TEXTTABLE`
- `TAB_NOINTP`

- BITFIELD\_TEXTTABLE
- LINEAR
- RAT\_FUNC
- IDENTICAL

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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`) **at the time when the RTE is generated.**]()

**[constr\_1155] Compatibility of CompuScales for client-server communication** [For client-server communication, the following rules apply **at the time when the RTE is generated:**

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.]()

**[constr\_1156] Relevance of “names” of CompuScales** [`CompuScales` which contribute to tabular conversion by having a `compuConst` are compatible **if and only if** the “names” of the `compuScales`, (namely `shortLabel`, `vt` and `symbol`, according to the priority rules communicated in [TPS\_SWCT\_01431]) are equal.

If the scale has no `compuConst`, “names” of `CompuScales` are not relevant for compatibility.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[TPS\_SWCT\_01842] Applicability of constraints of CompuScales** [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.]()

**[constr\_1176] Compatibility of CompuScales of category LINEAR and RAT\_FUNC** [CompuScales of category LINEAR and RAT\_FUNC are considered compatible **at the time when the RTE is generated** if they yield the same conversion.]()

For example,  $\frac{n_0+n_1*phys}{d_0+d_1*phys}$  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”** [Similar to [constr\_1176], a CompuScale where the category of the enclosing CompuMethod is set to IDENTICAL is – **at the time when the RTE is generated** – considered compatible to a CompuScale where the category of the enclosing CompuMethod is set to RAT\_FUNC or LINEAR if the following rule applies:

$$int = \frac{N_0+N_1*phys+N_i*phys^i}{D_0+D_1*phys+D_i*phys^i} = phys$$

]()

For example, this is the case for

$$N_0 \sim 0 \ \&\& \ D_0 \sim 1 \ \&\& \ N_1 \sim 1 \ \&\& \ D_1 \sim 0 \ \&\& \ N_i \sim D_i \sim 0 \ \forall i > 1.$$

Please note that the compatibility does not depend on the direction (`compuInternalToPhys` vs. `compuPhysToInternal`) of CompuMethods of category LINEAR.

#### 6.2.4.6 Compatibility of Record Layouts

**[constr\_1162] Compatibility of SwRecordLayouts** [Two SwRecordLayout definitions are compatible **at the time when the RTE is generated** if and only if all attributes **except**

- `shortName`
- `desc`
- `introduction`
- `longName`
- `adminData`
- `annotation`

are **identical**.]()

## 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.]()

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`, then 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` shall (after all indirections created by `ImplementationDataTypes` of `category TYPE_REFERENCE` are resolved) only be mapped/connected to an `ImplementationDataType` which also has `category VALUE`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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`).

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 shall (after all indirections created by `ImplementationDataTypes` of category TYPE\_REFERENCE are resolved) 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]).

The specific rules are documented in Table 6.1. This constraint shall be imposed **at the time when the contract phase generation is executed.**]()

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/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 shall (after all indirections created by `ImplementationDataTypes` of category TYPE\_REFERENCE are resolved) only be mapped/connected to an `ImplementationDataType` of category STRUCTURE.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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\_1662] Compatibility of `ApplicationRecordDataType` and `ImplementationDataType` that both represent an `Optional Element Structure`** [An `ApplicationRecordDataType` that represents an `Optional Element Structure` shall (after all indirections created by `ImplementationDataTypes` of category `TYPE_REFERENCE` are resolved) only be mapped/connected to an `ImplementationDataType` of category `STRUCTURE` that represents an `Optional Element Structure` if corresponding pairs of elements have the same value of the attribute `isOptional`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

6. **[constr\_1063] Compatibility of data types with category `BOOLEAN`** [An `ApplicationDataType` of category `BOOLEAN` shall (after all indirections created by `ImplementationDataTypes` of category `TYPE_REFERENCE` are resolved) only be mapped/connected to an `ImplementationDataType` of category `VALUE`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

7. **[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`

shall (after all indirections created by `ImplementationDataTypes` of category `TYPE_REFERENCE` are resolved) only be mapped/connected to an `ImplementationDataType` of category

- `STRUCTURE` or
- `ARRAY`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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 `SwRecordLayout`.

It is not required, to define `DataTypeMaps` for the sub-elements or both representations.

8. **[constr\_1066] Forbidden mappings to `ImplementationDataType`** [An `ApplicationDataType` shall never be mapped to

- an `ImplementationDataType` of category
  - UNION,
  - DATA\_REFERENCE, or
  - FUNCTION\_REFERENCE,
- or to an `ImplementationDataType` that contains `subElements` of category
  - UNION,
  - DATA\_REFERENCE, or
  - FUNCTION\_REFERENCE.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

	Array of uint8	Array of other
<code>ApplicationArrayDataType, arraySizeSemantics = fixedSize</code>	<code>ImplementationDataType</code> of category ARRAY, with <code>ImplementationDataTypeElement</code> with <code>arraySizeSemantics = fixedSize</code>	<code>ImplementationDataType</code> of category ARRAY, with <code>ImplementationDataTypeElement</code> with <code>arraySizeSemantics = fixedSize</code>
<code>ApplicationArrayDataType, arraySizeSemantics = variableSize</code>	<code>ImplementationDataType</code> of category ARRAY, with <code>ImplementationDataTypeElement</code> with <code>arraySizeSemantics = variableSize</code> or <code>Variable-Size Array Data Type</code>	<code>Variable-Size Array Data Type</code>

**Table 6.1: Rules for compatibility of old and new world variable-size arrays**

Concerning the `SwDataDefProps` of an `ApplicationDataType` instance or an `ImplementationDataType` instance which shall be mapped/connected on M1, we refer to the table shown in figure 5.43. 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 **at the time when the RTE is generated** if and only if one of the following conditions applies:

1. All 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 **at the time when the RTE is generated** 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, except for 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 **at the time when the RTE is generated** 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 6.2 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 **at the time when the RTE is generated** 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 **at the time when the RTE is generated** 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`.

**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.

]()

The table 6.2 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`.

#### 6.4.4 Compatibility of ParameterDataPrototype and VariableDataPrototype depending on PortInterface Type

Table 6.2 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 **at the time when the RTE is generated** where Table 6.2 contains the value “yes”.] ()

The following legend applies for the abbreviations used in table 6.2:

**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					
<code>PortInterface</code>			Prm			S/R		NvD
Interface Element			PDP			VDP		VDP
<code>SwImplPolicyEnum</code>			<code>fixed</code>	<code>const</code>	<code>standard</code>	<code>standard</code>	<code>queued</code>	<code>standard</code>
Prm	PDP	<code>fixed</code>	yes	yes	yes	yes	no	yes
		<code>const</code>	no	yes	yes	yes	no	yes
		<code>standard</code>	no	no	yes	yes	no	yes
S/R	VDP	<code>standard</code>	no	no	no	yes	no	yes
		<code>queued</code>	no	no	no	no	yes	no
NvD	VDP	<code>standard</code>	no	no	no	yes	no	yes

**Table 6.2: 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 `invalidationPolicys` have the same value.

This rule shall be imposed **at the time when the RTE is generated.** ]()

**[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 **at the time when the RTE is generated** 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 **at the time when the RTE is generated** 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 **at the time when the RTE is generated** 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 **at the time when the RTE is generated** 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 **at the time when the RTE is generated** if and only if one of the following conditions applies:

1. All 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 **at the time when the RTE is generated**:

- **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` **at the time when the RTE is generated** 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 `shortName` 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 **at the time when the RTE is generated** 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 **at the time when the RTE is generated** if and only if one of the following conditions applies:

1. All 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 considered compatible **at the time when the RTE is generated** 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 identical values of attribute `diagArgIntegrity` or the attribute `diagArgIntegrity` does not exist on both sides.
4. They have the same `shortName` (again allows for mapping in `PortInterfaces`).
5. 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`.

]0

## 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 **at the time when the RTE is generated** 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.

]0

### 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 **at the time when the RTE is generated** 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 **at the time when the RTE is generated** if and only if

1. For **at least one** `ClientServerOperation` defined in the context of the `ClientServerInterface` of the provided outer `PortPrototype` a compatible `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 **at the time when the RTE is generated** 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 **at the time when the RTE is generated** if and only if all 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 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 **at the time when the RTE is generated** 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 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 **at the time when the RTE is generated** 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.

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 `PassThroughSwConnector` 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.

This constraint is associated with two possible imposition times:

- at the time when the RTE is generated
- at the time when the creation of the `CompositionSwComponentType` is finished

]()

Table 6.2 defines which `PortInterface` elements are compatible depending on the kind of `PortInterface` and the `swImplPolicy` attributes of the `PortInterface` elements.

### 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` typed by 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 6.2 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.

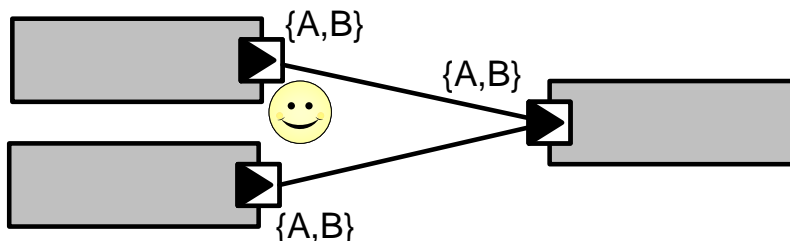


Figure 6.1: legal n:1 communication

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.

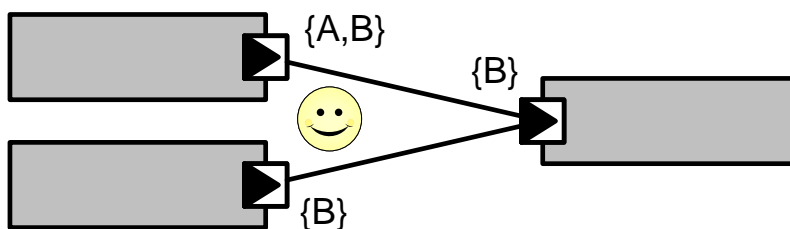


Figure 6.2: legal n:1 communication

### 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.

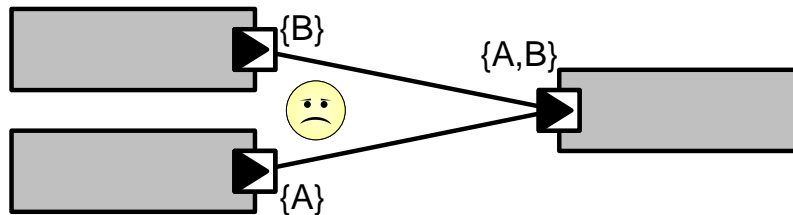


Figure 6.3: illegal n:1 communication

### 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`.

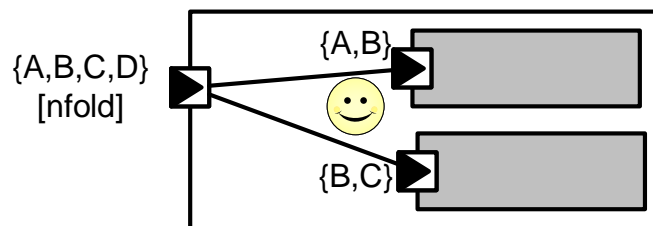


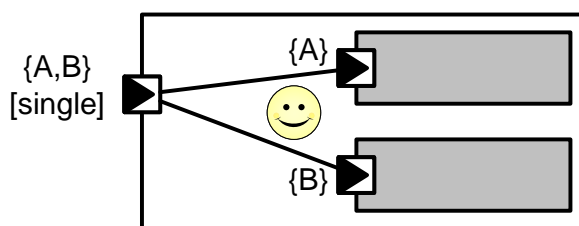
Figure 6.4: Legal split of delegation connector

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`.

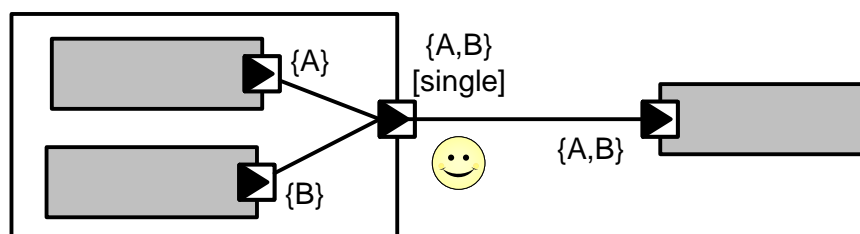
In the next example, the `RPortPrototype` of the `CompositionSwComponentType` contains the superset of `dataElements` {A, B}. The two `RPortPrototypes` of the `SwComponentPrototypes` contain *different* subsets, i.e. {A} and {B}.



**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 `PPortPrototype` 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}.



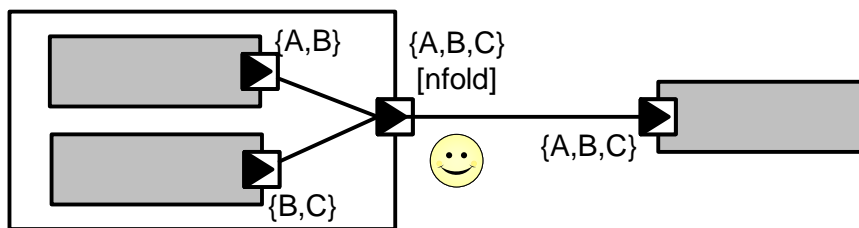
**Figure 6.6: Legal merge of delegation connector**

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}.

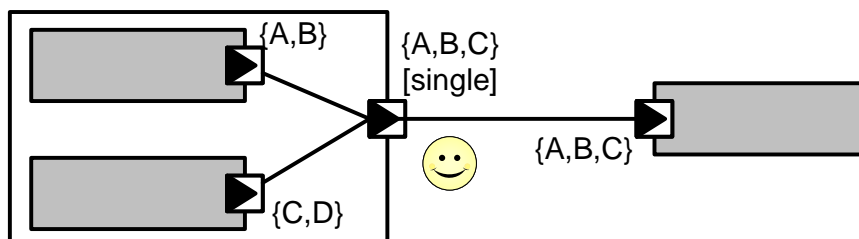
The *PPortPrototypes* of the *SwComponentPrototypes* in turn contain subsets of *dataElements*, i.e. {A, B} and {B, C}. In this case the resulting communication pattern on the VFB for {B} would be n:1.



**Figure 6.7: Legal merge of delegation connector**

This would require the value of the attribute *signalFan* of *DelegatedPortAnnotation* to be set to *ifold*. 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*.

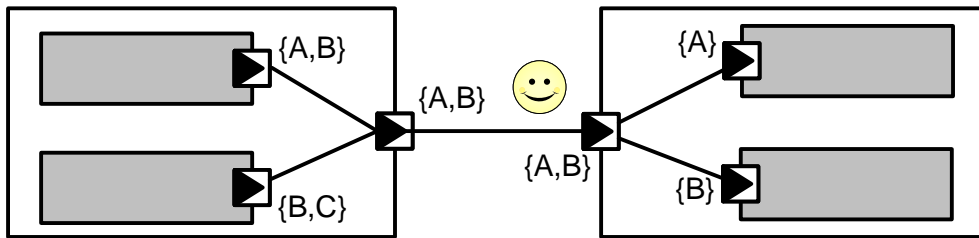


Figure 6.9: Valid delegation of **SwConnectors** that goes end-to-end

### 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*

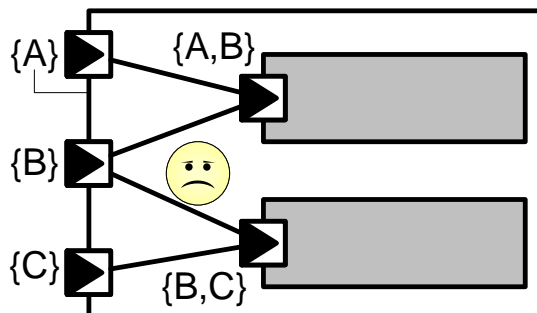


Figure 6.10: Illegal split of delegation connector

In the next example compatibility is also not fulfilled because the required **dataElement** **{E}** is not provided by the delegation **RPortPrototype**.

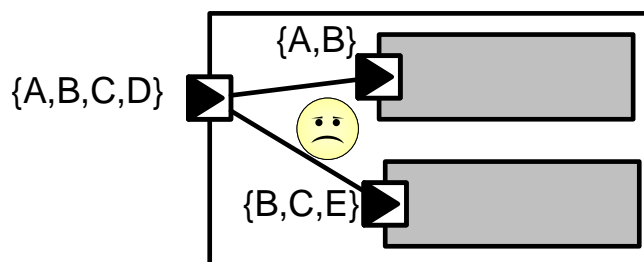
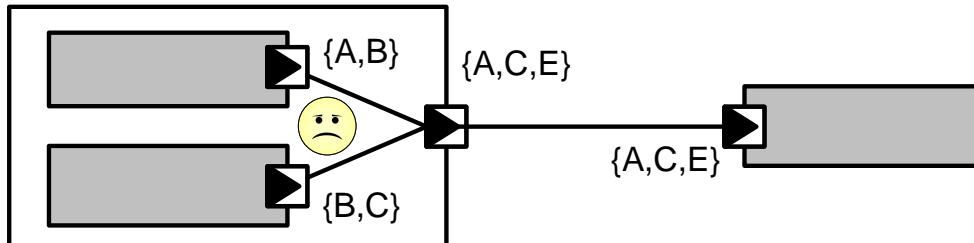


Figure 6.11: Illegal split of delegation connector

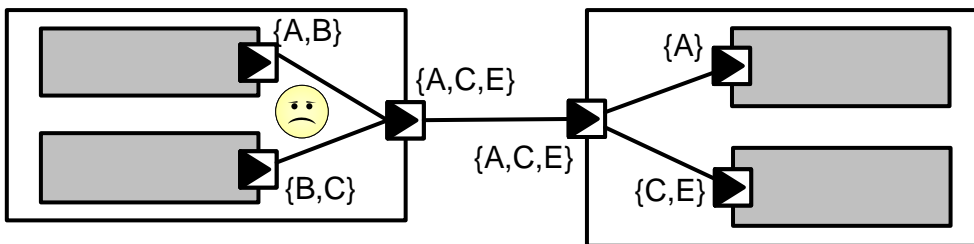
An incompatible merge of *DelegationSwConnectors* is sketched in Figure 6.12. In this case the *dataElement* {E} is *not* provided by one of the *PPortPrototypes* owned by the *SwComponentPrototypes* inside the *CompositionSwComponentType*.



**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 «atpSplittable» is to allow for the development of SwcInternalBehavior in a later process step (e.g. after the VFB view has been completed).

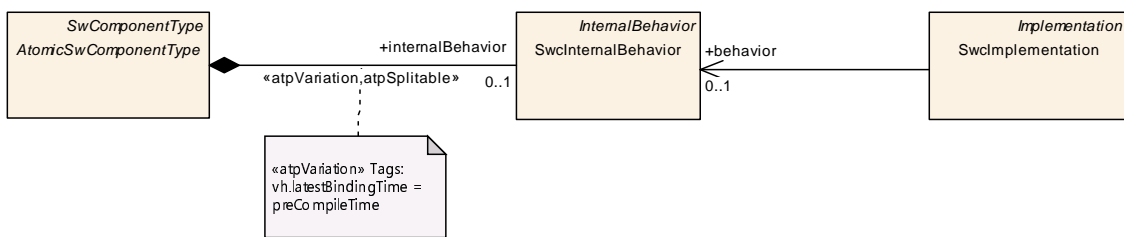


Figure 7.1: The “big picture” of SwcInternalBehavior

<b>Class</b>	<i>InternalBehavior</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
<b>Note</b>	Common base class (abstract) for the internal behavior of both software components and basic software modules/clusters.			
<b>Base</b>	<i>ARObject</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Subclasses</b>	<i>BswInternalBehavior</i> , <i>SwcInternalBehavior</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
constant Memory	<a href="#">ParameterData</a> <a href="#">Prototype</a>	*	aggr	Describes a read only memory object containing characteristic value(s) implemented by this Internal Behavior.  The shortName of ParameterDataPrototype has to be equal to the 'C' identifier of the described constant.  The characteristic value(s) might be shared between Sw ComponentPrototypes 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.







Class	<i>InternalBehavior</i> (abstract)			
				△ <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=constantMemory.shortName, constantMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
constantValue Mapping	<a href="#">ConstantSpecification MappingSet</a>	*	ref	Reference to the ConstantSpecificationMapping to be applied for the particular InternalBehavior <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=constantValueMapping
dataType Mapping	<a href="#">DataTypeMappingSet</a>	*	ref	Reference to the DataTypeMapping to be applied for the particular InternalBehavior <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=dataTypeMapping
exclusiveArea	<a href="#">ExclusiveArea</a>	*	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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=exclusiveArea.shortName, exclusiveArea.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
exclusiveArea NestingOrder	<a href="#">ExclusiveAreaNesting Order</a>	*	aggr	This represents the set of ExclusiveAreaNestingOrder owned by the InternalBehavior. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=exclusiveAreaNestingOrder.shortName, exclusiveAreaNestingOrder.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
staticMemory	<a href="#">VariableDataPrototype</a>	*	aggr	Describes a read and writeable static memory object representing measurement variables implemented by this software component. The term "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 component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=staticMemory.shortName, staticMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

**Table 7.1: InternalBehavior**

<b>Class</b>	<b>SwcInternalBehavior</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
<b>Note</b>	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.			
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">InternalBehavior</a> , <a href="#">Multilanguage</a> , <a href="#">Referrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arTypedPerInstanceMemory	<a href="#">VariableDataPrototype</a>	*	aggr	<p>Defines an AUTOSAR typed memory-block that needs to be available for each instance of the SW-component.</p> <p>This is typically only useful if supportsMultipleInstantiation is set to "true" or if the component defines NVRAM access via permanent blocks.</p> <p>The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the software component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=arTypedPerInstanceMemory.shortName, arTypedPerInstanceMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
event	<a href="#">RTEEvent</a>	*	aggr	<p>This is a RTEEvent specified for the particular SwcInternalBehavior.</p> <p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=event.shortName, event.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
exclusiveAreaPolicy	<a href="#">SwcExclusiveAreaPolicy</a>	*	aggr	<p>Options how to generate the ExclusiveArea related APIs. When no SwcExclusiveAreaPolicy is specified for an ExclusiveArea the default values apply.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=exclusiveAreaPolicy, exclusiveAreaPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
explicitInterRunnableVariable	<a href="#">VariableDataPrototype</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=explicitInterRunnableVariable.shortName, explicitInterRunnableVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>





Class	SwcInternalBehavior			
handleTerminationAndRestart	<a href="#">HandleTerminationAndRestartEnum</a>	0..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.
implicitInterRunnableVariable	<a href="#">VariableDataPrototype</a>	*	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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=implicitInterRunnableVariable.shortName, implicitInterRunnableVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedDataTypeSet	<a href="#">IncludedDataTypeSet</a>	*	aggr	The includedDataTypeSet is used by a software component for its implementation. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=includedDataTypeSet
includedModeDeclarationGroupSet	<a href="#">IncludedModeDeclarationGroupSet</a>	*	aggr	This aggregation represents the included ModeDeclarationGroups <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=includedModeDeclarationGroupSet
instantiationDataDefProps	<a href="#">InstantiationDataDefProps</a>	*	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 Port Prototypes and component local memories like "per InstanceParameter" or "arTypedPerInstanceMemory". <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=instantiationDataDefProps, instantiationDataDefProps.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstanceMemory	<a href="#">PerInstanceMemory</a>	*	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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=perInstanceMemory.shortName, perInstanceMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime





Class	SwcInternalBehavior			
perInstanceParameter	<a href="#">ParameterData Prototype</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=perInstanceParameter.shortName, perInstanceParameter.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime</p>
portAPIOption	<a href="#">PortAPIOption</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=portAPIOption, portAPIOption.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime</p>
runnable	<a href="#">RunnableEntity</a>	*	aggr	<p>This is a RunnableEntity specified for the particular Swc InternalBehavior.</p> <p>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 Port Prototypes using DataReceivedEvents or due to different scheduling needs of algorithms.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=runnable.shortName, runnable.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime</p>
serviceDependency	<a href="#">SwcService Dependency</a>	*	aggr	<p>Defines the requirements on AUTOSAR Services for a particular item.</p> <p>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.</p> <p>The SwcServiceDependency owned by an SwcInternal Behavior 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 &lt;&lt;atp Splitable&gt;&gt;.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=serviceDependency.shortName, serviceDependency.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime</p>





Class	SwcInternalBehavior			
shared Parameter	<a href="#">ParameterData Prototype</a>	*	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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=sharedParameter.shortName, sharedParameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
supports Multiple Instantiation	Boolean	0..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).
variationPoint Proxy	<a href="#">VariationPointProxy</a>	*	aggr	Proxy of a variation points in the C/C++ implementation. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=variationPointProxy.shortName

**Table 7.2: SwcInternalBehavior**

**[constr\_1934] Existence of attribute [SwcInternalBehavior.handleTerminationAndRestart](#)** [For each [SwcInternalBehavior](#), attribute [handleTerminationAndRestart](#) shall exist **at the time when the RTE is generated.**]()

**[constr\_1935] Existence of attribute [SwcInternalBehavior.supportsMultipleInstantiation](#)** [For each [SwcInternalBehavior](#), attribute [supportsMultipleInstantiation](#) shall exist **at the time when the contract phase generation is executed.**]()

Enumeration	HandleTerminationAndRestartEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior
Note	Controls the behavior of an AtomicSwComponentType with respect to stop and restart.
Literal	<b>Description</b>
canBeTerminated	Supports termination. <b>Tags:</b> atp.EnumerationLiteralIndex=0
canBeTerminated AndRestarted	Supports termination and restarting. <b>Tags:</b> atp.EnumerationLiteralIndex=1
noSupport	Stop and restart is not supported at all. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 7.3: HandleTerminationAndRestartEnum**

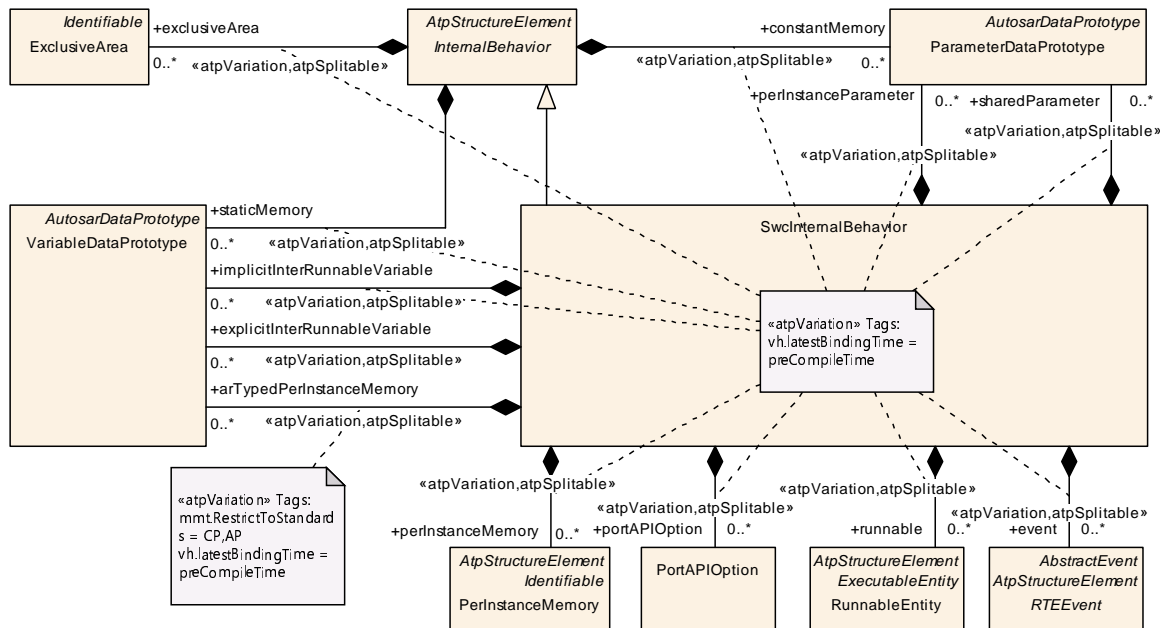


Figure 7.2: SwcInternalBehavior

## 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** [RunnableEntities 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 or else (in rare cases) for execution in ISR context.](RS\_SWCT\_03040, RS\_SWCT\_00070, RS\_SWCT\_00090, RS\_SWCT\_03050)

**[TPS\_SWCT\_01097] CompositionSwComponentType cannot have RunnableEntities** [It is intentionally not possible for CompositionSwComponentType 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\_03040, RS\_SWCT\_00070, RS\_SWCT\_00090, RS\_SWCT\_03050)

This correlation is depicted in Figure 7.4.

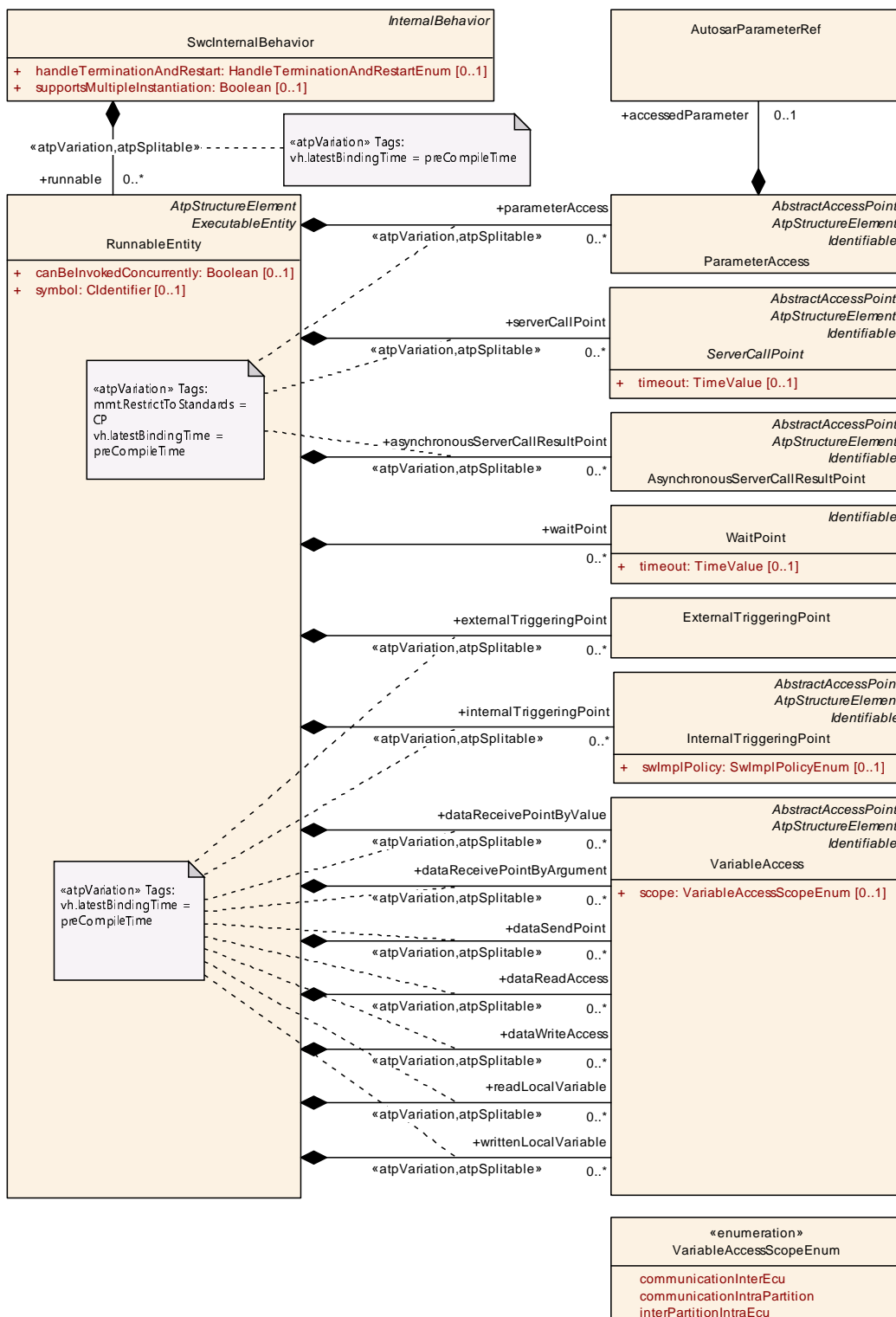
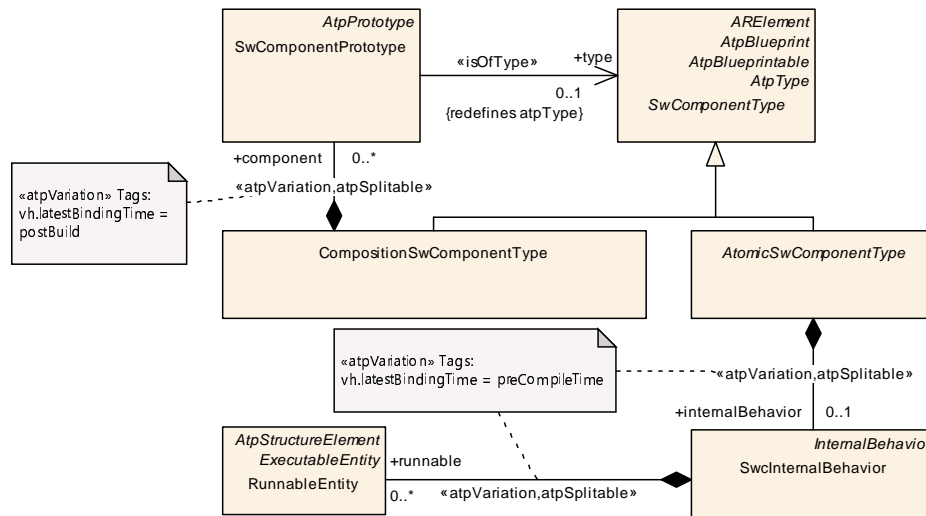


Figure 7.3: Details of RunnableEntity

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.



**Figure 7.4: Only AtomicSwComponentTypes may have RunnableEntities**

<b>Class</b>	<b>RunnableEntity</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
<b>Note</b>	A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponent Type 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.			
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, ExecutableEntity, Identifiable, Multilanguage Referrable, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
argument (ordered)	RunnableEntity Argument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.
asynchronous ServerCall ResultPoint	AsynchronousServer CallResultPoint	*	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.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=asynchronousServerCallResultPoint.short Name, asynchronousServerCallResultPoint.variation Point.shortLabel vh.latestBindingTime=preCompileTime
canBelnvoked Concurrently	Boolean	0..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 AtomicSwComponent Type). This implies that it is the responsibility of the implementation of the RunnableEntity to take care of this form of concurrency.







Class	RunnableEntity			
dataRead Access	<a href="#">VariableAccess</a>	*	aggr	<p>RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=dataReadAccess.shortName, dataReadAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
dataReceive PointBy Argument	<a href="#">VariableAccess</a>	*	aggr	<p>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.</p> <p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=dataReceivePointByArgument.shortName, dataReceivePointByArgument.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
dataReceive PointByValue	<a href="#">VariableAccess</a>	*	aggr	<p>RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=dataReceivePointByValue.shortName, dataReceivePointByValue.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
dataSendPoint	<a href="#">VariableAccess</a>	*	aggr	<p>RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=dataSendPoint.shortName, dataSendPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>





Class	RunnableEntity			
dataWrite Access	<a href="#">VariableAccess</a>	*	aggr	<p>RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=dataWriteAccess.shortName, dataWriteAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
external TriggeringPoint	<a href="#">ExternalTriggeringPoint</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=externalTriggeringPoint.ident.shortName, externalTriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
internal TriggeringPoint	<a href="#">InternalTriggeringPoint</a>	*	aggr	<p>The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=internalTriggeringPoint.shortName, internalTriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
modeAccess Point	<a href="#">ModeAccessPoint</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=modeAccessPoint.ident.shortName, modeAccessPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
modeSwitch Point	<a href="#">ModeSwitchPoint</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=modeSwitchPoint.shortName, modeSwitchPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>





Class	RunnableEntity			
parameter Access	<a href="#">ParameterAccess</a>	*	aggr	<p>The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a Parameter DataPrototype which may either be local or within a Port Prototype.</p> <p>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 Parameter Access (points) in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=parameterAccess.shortName, parameter Access.variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
readLocal Variable	<a href="#">VariableAccess</a>	*	aggr	<p>The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable.</p> <p>The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of read LocalVariable (points) in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=readLocalVariable.shortName, readLocal Variable.variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
serverCallPoint	<a href="#">ServerCallPoint</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=serverCallPoint.shortName, serverCall Point.variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
symbol	CIdentifier	0..1	attr	<p>The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.</p>
waitPoint	<a href="#">WaitPoint</a>	*	aggr	<p>The WaitPoint associated with the RunnableEntity.</p>
writtenLocal Variable	<a href="#">VariableAccess</a>	*	aggr	<p>The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable.</p> <p>The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of written LocalVariable (points) in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=writtenLocalVariable.shortName, written LocalVariable.variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>

**Table 7.4: 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`.] (*RS\_SWCT\_03040*)

**[constr\_1936] Existence of attribute `RunnableEntity.symbol`** [For each `RunnableEntity`, attribute `symbol` shall exist **at the time when the contract phase generation is executed.**] ()

**[TPS\_SWCT\_01303] `symbol` attribute describes the `RunnableEntity`'s entry point** [The `RunnableEntity.symbol` attribute is describing the `RunnableEntity`'s entry point.] (*RS\_SWCT\_03040*)

The implication `RunnableEntity.symbol` on the uniqueness of symbols in the scope of one `EcuInstance` is described in [constr\_2025] [10].

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 [6]. 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.

This rule shall be imposed **at the time when the contract phase generation is executed.**] ()

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.

Please note further that there are legitimate use cases for software-components without any `RunnableEntities`, e.g. in following situations:

- An `NvBlockSwComponentType` does not require any `RunnableEntity` if there is no need to proxy any `PortPrototype` typed by either of the `ClientServerInterfaces` `NvMService` or `NvMAdmin`.
- A `ServiceSwComponentType` runs in a reduced configuration and does not have to offer any `PortPrototype` to any service-using application software-component.
- A software-component is configured in a reduced configuration where none of the functionality is selected. In this case, it's simpler to keep the empty software-component instead of adding further `VariationPoints` at many other elements, e.g. `SwComponentPrototype`.

On top of that, a variation-based approach would require the conditional existence of other `ARElements` which are not yet supported, e.g. a `SwcImplementation` that references the `AtomicSwComponentType`.

## 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`.] ([RS\\_SWCT\\_03040](#))

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 [31]). 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).] ([RS\\_SWCT\\_03040](#))

**[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.] ([RS\\_SWCT\\_03040](#))

For example: The internal behavior of an `AtomicSwComponentType` *MyComponentType* describes a `RunnableEntity` *R1* which should be enabled when an `operation` on a client-server `PPortPrototype` 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` anymore, 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.] (*RS\_SWCT\_03040*)

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 `RunnableEntities` 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, `RunnableEntities` 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 `RunnableEntities` 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`.

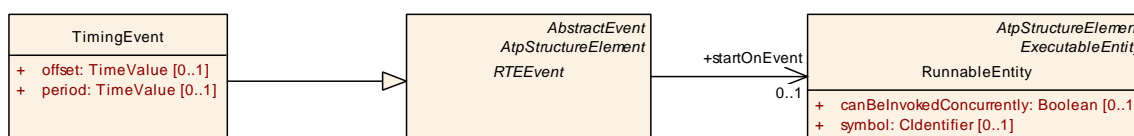


Figure 7.5: Periodic activation of RunnableEntities

**[TPS\_SWCT\_01519] RTE executes certain `RunnableEntity` periodically** [If the `SwcInternalBehavior` of an `AtomicSwComponentType` requires that the RTE executes certain `RunnableEntities` periodically, the description needs to define a `TimingEvent` with the desired period.

This `TimingEvent` then contains a reference to the `RunnableEntity` that needs to be executed with this period.] (*RS\_SWCT\_03040*)

**[constr\_2031] Period of `TimingEvent` shall be greater than 0** [The value of the attribute period of `TimingEvent` shall be greater than 0 **at the time when the RTE is generated.**]()

Note that it is possible to override the attribute `period` on the level of instantiation. See [TPS\_SWCT\_02507] for more details.

<b>Class</b>	TimingEvent			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is used to start RunnableEntities that shall be executed periodically.			
<b>Base</b>	ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, Multilanguage Referrable, RTEEvent, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
offset	TimeValue	0..1	attr	The value makes an assumption about the time offset of the first activation of the RunnableEntity triggered by the mapped TimingEvent relative to the periodic activation of the time base of this TimingEvent. Unit: second.
period	TimeValue	0..1	attr	Period of timing event in seconds. The value of this attribute shall be greater than zero.

Table 7.5: TimingEvent

[constr\_1937] Existence of attribute `TimingEvent.period` [For each `TimingEvent`, attribute `period` shall exist at the time when the RTE is generated.]()

[constr\_1622] Value of `TimingEvent.offset` vs. `TimingEvent.period` [If a value is defined for attribute `TimingEvent.offset` then this value shall be greater than 0 and less or equal than the value of attribute `TimingEvent.period` of the respective `TimingEvent` at the time when the RTE is generated.]()

The motivation for the existence of [constr\_1622] is that the mapped `TimingEvent` could not be implemented with the given `period` if the activation `offset` is greater than the period of the `TimingEvent`.

## 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 7.6.] (*RS\_SWCT\_03040*)

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.



<code>supportsMultipleInstantiation</code>	<code>canBeInvokedConcurrently</code>	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.
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 7.6: `supportsMultipleInstantiation` vs. `canBeInvokedConcurrently`**

### 7.2.4.2 Reentrancy and “Library Functions”

Note that all code that is called by different `RunnableEntity`s (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` that trigger the same `RunnableEntity` are only supported if all referenced `ClientServerOperations` would result in the same `RunnableEntity` signature for the server `RunnableEntity`.] (*RS\_SWCT\_03040*)

**[constr\_2000] Compatibility of `ClientServerOperations` triggering the same `RunnableEntity`** [The `ClientServerOperations` are considered compatible **at the time when the contract phase generation is executed** 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 or both are typed by "new-world" `Variable-Size Array Data Types` where the data types of the array elements are compatible (but the array sizes may differ).
- and the respective values of `PortAPIOption.errorHandling` are identical.

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.

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`.] (*RS\_SWCT\_03040*)

#### 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** `RunnableEntity`s of Category 1 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 APIs.

**Category 1B** `RunnableEntity`s are additionally allowed to invoke a server, to use explicit APIs, to issue triggers, to switch modes and to use `ExclusiveAreas`.

**Category 2** In contrast to Category 1, `RunnableEntity`s, `RunnableEntity`s of Category 2 always aggregate at least one `WaitPoint`<sup>1</sup>. Typically, such a

<sup>1</sup>Category 2 `RunnableEntity`s usually have to be mapped to *Extended Tasks*, because only extended tasks provide the task state WAITING.

`RunnableEntity` implements an internal loop where one iteration through the loop is triggered whenever a `WaitPoint` is resolved.

]([RS\\_SWCT\\_03040](#))

For more details regarding details of the modeling of meta-class `RunnableEntity`, please refer to Figure 7.3.

The resource need of a `RunnableEntity` in its later integration usually depends on the used features of the Runtime Environment.

In the AUTOSAR Methodology, the ECU integrator is required to map various `RunnableEntities` to a limited amount of OS Tasks in a specific order.

The `RunnableEntity` categories are useful to indicate the later integration effort and resource need on the basis of the `RunnableEntity`'s design.

Thereby Category 1A `RunnableEntities` do not utilize RTE features which are blocking or delaying the execution of the `RunnableEntity`. As long as the `RunnableEntity` implementation guarantees stable execution times it's rather simple and reliable to integrate them in a calculation chain.

As opposed to Category 1A, the scheduling behavior of Category 2 `RunnableEntities` at runtime depends on the interaction with the interfaces of the enclosing software component.

For instance, the suspend-times of the OS Task, where the `RunnableEntity` is mapped to, may depend on actual data reception and/or occurrence of timeouts.

Moreover, it's usually not possible to map more than one `RunnableEntity` to an OS Task when the `RunnableEntity` implements an infinite internal loop, triggered whenever a `WaitPoint` is resolved.

In case of Category 1B, additional side conditions impact the schedule behavior and required OS features. For instance, a server call might be simply implemented as direct function call in case of intra partition communication **or** might require a more complex implementation in case of inter-ECU communication.

#### 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<sup>2</sup>.]()

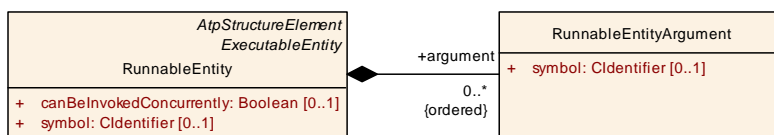
**[constr\_1164] Number of arguments owned by a `RunnableEntity`** [If a given `RunnableEntity` owns `RunnableEntityArguments` in the role `argument`, then the number of these `RunnableEntityArguments` shall be identical to the number of applicable `portArgValues` of the `PortAPIOption` that references 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` **at the time when the contract phase generation is executed.**]()

**[constr\_1165] Applicability of `RunnableEntityArgument`** [The existence of a `RunnableEntityArgument` is limited to `RunnableEntities` triggered by a `ClientServerOperation`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[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`.] (*RS\_SWCT\_03040*)



**Figure 7.6: Arguments of a `RunnableEntity`**

<b>Class</b>	<b>RunnableEntityArgument</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RunnableEntity			
<b>Note</b>	This meta-class represents the ability to provide specific information regarding the arguments to a <code>RunnableEntity</code> .			
<b>Base</b>	<code>ARObject</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>



<sup>2</sup>as the arguments are **ordered** they do not need to be `Referrable` in order to be able to identify individual `arguments`



Class	RunnableEntityArgument			
symbol	CIdentifier	0..1	attr	This represents the symbol to be generated into the actual signature on the level of the C programming language.

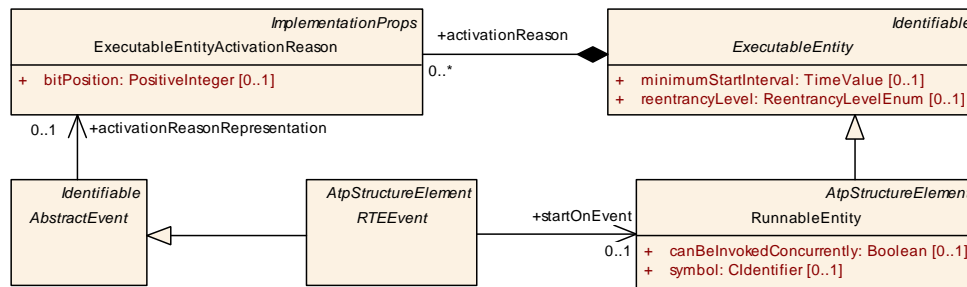
**Table 7.7: RunnableEntityArgument**

**[constr\_1938] Existence of attribute `RunnableEntityArgument.symbol`** [For each `RunnableEntityArgument`, attribute `symbol` shall exist **at the time when the contract phase generation is executed.**]()

### 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\\_03040](#), [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 **at the time when the contract phase generation is executed.**]()

**[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 at the time when the contract phase generation is executed.]()

**[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 at the time when the RTE is generated.]()

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.

<b>Class</b>	<i>ExecutableEntity</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
<b>Note</b>	Abstraction of executable code.			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable			
<b>Subclasses</b>	BswModuleEntity, RunnableEntity			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.
canEnter	ExclusiveArea	*	ref	This means that the executable entity can enter/leave the referenced exclusive area through explicit API calls.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
canEnterExclusiveArea	ExclusiveArea	*	ref	This means that the executable entity can enter/leave the referenced exclusive area through explicit API calls.  <b>Tags:</b> atp.Status=obsolete
exclusiveAreaNestingOrder	ExclusiveAreaNestingOrder	*	ref	This represents the set of ExclusiveAreaNestingOrders recognized by this ExecutableEntity.
minimumStartInterval	TimeValue	0..1	attr	Specifies the time in seconds by which two consecutive starts of an ExecutableEntity are guaranteed to be separated.
reenetrancyLevel	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.





Class	ExecutableEntity (abstract)			
runInside	<a href="#">ExclusiveArea</a>	*	ref	The executable entity runs completely inside the referenced exclusive area. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
runInside ExclusiveArea	<a href="#">ExclusiveArea</a>	*	ref	The executable entity runs completely inside the referenced exclusive area. <b>Tags:</b> atp.Status=obsolete
swAddrMethod	<a href="#">SwAddrMethod</a>	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 7.8: ExecutableEntity**

Class	ExecutableEntityActivationReason			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	This meta-class represents the ability to define the reason for the activation of the enclosing Executable Entity.			
Base	<a href="#">ARObject</a> , <a href="#">ImplementationProps</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
bitPosition	PositiveInteger	0..1	attr	This attribute allows for defining the position of the enclosing ExecutableEntityActivationReason in the activation vector.

**Table 7.9: ExecutableEntityActivationReason**

**[constr\_1939] Existence of attribute [ExecutableEntityActivationReason.bitPosition](#)** [For each [ExecutableEntityActivationReason](#), attribute [bitPosition](#) shall exist **at the time when the contract phase generation is executed.**]

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 `RunnableEntity`s 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 `RunnableEntity`s 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 `RunnableEntity`s 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` **at the time when the RTE is generated.**] ()

Rationale: a `WaitPoint` may indefinitely defer the completion of the `RunnableEntity`s triggered by an `InitEvent` and therefore contradict the semantics of the `RunnableEntity`.

**[constr\_1258] Value of minimumStartInterval for RunnableEntitys triggered by an InitEvent** [The value of the attribute `ExecutableEntity.minimumStartInterval` for a `RunnableEntity`s that is triggered by an `InitEvent` shall always be set to 0 **at the time when the RTE is generated.**] ()

Rationale: it does not make sense to talk about intervals of activating `RunnableEntity`s 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` **at the time when the RTE is generated.**] ()

Rationale: as mentioned before `WaitPoints` shall not be aggregated by a `RunnableEntity`s 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` **at the time when the RTE is generated.**] ()

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](#) [An [RTEEvent](#) defines:

- what the trigger for the occurrence of that [RTEEvent](#) is
- whether specific [ModeDeclarations](#) disable the processing of this [RTEEvent](#)
- which [RunnableEntity](#) shall be started when this [RTEEvent](#) occurs.

]([RS\\_SWCT\\_03040](#))

<b>Class</b>	<a href="#">AbstractEvent</a> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
<b>Note</b>	This meta-class represents the abstract ability to model an event that can be taken to implement application software or basic software in AUTOSAR.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">BswEvent</a> , <a href="#">RTEEvent</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
activationReasonRepresentation	<a href="#">ExecutableEntity</a> <a href="#">ActivationReason</a>	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 7.10: AbstractEvent**

<b>Class</b>	<a href="#">RTEEvent</a> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	Abstract base class for all RTE-related events			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AbstractEvent</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">AsynchronousServerCallReturnsEvent</a> , <a href="#">BackgroundEvent</a> , <a href="#">DataReceiveErrorEvent</a> , <a href="#">DataReceivedEvent</a> , <a href="#">DataSendCompletedEvent</a> , <a href="#">DataWriteCompletedEvent</a> , <a href="#">ExternalTriggerOccurredEvent</a> , <a href="#">InitEvent</a> , <a href="#">InternalTriggerOccurredEvent</a> , <a href="#">ModeSwitchedAckEvent</a> , <a href="#">OperationInvokedEvent</a> , <a href="#">OsTaskExecutionEvent</a> , <a href="#">SwcModeManagerErrorEvent</a> , <a href="#">SwcModeSwitchEvent</a> , <a href="#">TimingEvent</a> , <a href="#">TransformerHardErrorEvent</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
disabledMode	<a href="#">ModeDeclaration</a>	*	iref	Reference to the Modes that disable the Event. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=disabledMode.contextPort, disabledMode.contextModeDeclarationGroupPrototype, disabledMode.targetModeDeclaration <b>InstanceRef implemented by:</b> <a href="#">RModelNAtomicSwcInstanceRef</a>
startOnEvent	<a href="#">RunnableEntity</a>	0..1	ref	The referenced RunnableEntity starts when the corresponding RTEEvent is raised.

**Table 7.11: RTEEvent**

<b>Class</b>	<b>AsynchronousServerCallReturnsEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when an asynchronous server call is finished.			
<b>Base</b>	<i>ARObject</i> , <i>AbstractEvent</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>Multilanguage Referrable</i> , <i>RTEEvent</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
eventSource	<a href="#">AsynchronousServerCallResultPoint</a>	0..1	ref	The referenced <a href="#">AsynchronousServerCallResultPoint</a> raises this <a href="#">AsynchronousServerCallReturnsEvent</a> when the asynchronous server call returns.

**Table 7.12: AsynchronousServerCallReturnsEvent**

**[constr\_1940] Existence of attribute [AsynchronousServerCallReturnsEvent.eventSource](#)** [For each [AsynchronousServerCallReturnsEvent](#), attribute [eventSource](#) shall exist **at the time when the contract phase generation is executed**.]()

<b>Class</b>	<b>DataSendCompletedEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when the referenced explicit data element has been sent or an error occurred.			
<b>Base</b>	<i>ARObject</i> , <i>AbstractEvent</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>Multilanguage Referrable</i> , <i>RTEEvent</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
eventSource	<a href="#">VariableAccess</a>	0..1	ref	The referenced <a href="#">VariableAccess</a> raises this <a href="#">DataSendCompletedEvent</a> when the explicit write access was successful or an error occurred.

**Table 7.13: DataSendCompletedEvent**

**[constr\_1941] Existence of attribute [DataSendCompletedEvent.eventSource](#)** [For each [DataSendCompletedEvent](#), attribute [eventSource](#) shall exist **at the time when the contract phase generation is executed**.]()

<b>Class</b>	<b>DataWriteCompletedEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when an implicit write access was successful or an error occurred.			
<b>Base</b>	<i>ARObject</i> , <i>AbstractEvent</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>Multilanguage Referrable</i> , <i>RTEEvent</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
eventSource	<a href="#">VariableAccess</a>	0..1	ref	The referenced <a href="#">VariableAccess</a> raises this <a href="#">DataWriteCompletedEvent</a> when the implicit write access was successful or an error occurred.

**Table 7.14: DataWriteCompletedEvent**

**[constr\_1942] Existence of attribute [DataWriteCompletedEvent.eventSource](#)** [For each [DataWriteCompletedEvent](#), attribute [eventSource](#) shall exist **at the time when the contract phase generation is executed**.]()

<b>Class</b>	<b>DataReceivedEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when the referenced data element is received.			
<b>Base</b>	ARObject, <a href="#">AbstractEvent</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">RTEEvent</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
data	<a href="#">VariableDataPrototype</a>	0..1	iref	The referenced VariableDataPrototype raises this DataReceivedEvent when the data has been received.  <b>InstanceRef implemented by:</b> <a href="#">RVariableInAtomicSwcInstanceRef</a>

**Table 7.15: DataReceivedEvent**

**[constr\_1943] Existence of attribute [DataReceivedEvent.data](#)** [For each [DataReceivedEvent](#), attribute [data](#) shall exist **at the time when the contract phase generation is executed.**.]()

<b>Class</b>	<b>DataReceiveErrorEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when the Com layer detects and notifies an error concerning the reception of the referenced VariableDataPrototype.			
<b>Base</b>	ARObject, <a href="#">AbstractEvent</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">RTEEvent</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
data	<a href="#">VariableDataPrototype</a>	0..1	iref	The referenced VariableDataPrototype raises this DataReceiveErrorEvent when there was an error during the reception.  <b>InstanceRef implemented by:</b> <a href="#">RVariableInAtomicSwcInstanceRef</a>

**Table 7.16: DataReceiveErrorEvent**

**[constr\_1944] Existence of attribute [DataReceiveErrorEvent.data](#)** [For each [DataReceiveErrorEvent](#), attribute [data](#) shall exist **at the time when the contract phase generation is executed.**.]()

<b>Class</b>	<b>OperationInvokedEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when the ClientServerOperation referenced in OperationInvokedEvent.operation shall be invoked.			
<b>Base</b>	ARObject, <a href="#">AbstractEvent</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">RTEEvent</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
operation	<a href="#">ClientServerOperation</a>	0..1	iref	This represents the ClientServerOperation which shall be invoked.  <b>InstanceRef implemented by:</b> <a href="#">POperationInAtomicSwcInstanceRef</a>

**Table 7.17: OperationInvokedEvent**

**[constr\_1945] Existence of attribute `OperationInvokedEvent.operation`** [For each `OperationInvokedEvent`, attribute `operation` shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1523] No mode disabling for `OperationInvokedEvents`** [An `OperationInvokedEvent` shall not have a reference to a `ModeDeclaration` in the role `disabledMode` **at the time when the RTE is generated.**]()

Rationale for the existence of [constr\_1523]:

The RTE does not support the disabling of server `RunnableEntity`s by modes. Instead, the server shall respond with an explicit error code if the execution of the server operation is not possible in specific side conditions.

For more explanation about the semantics of meta-class `TimingEvent`, please refer to section 7.2.3.

<b>Class</b>	<b>BackgroundEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is used to start <code>RunnableEntity</code> s that are supposed to be executed in the background.			
<b>Base</b>	<code>ARObject</code> , <code>AbstractEvent</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>RTEEvent</code> , <code>Referrable</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 7.18: BackgroundEvent**

<b>Class</b>	<b>SwcModeSwitchEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when the specified mode change occurs.			
<b>Base</b>	<code>ARObject</code> , <code>AbstractEvent</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>RTEEvent</code> , <code>Referrable</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
activation	<code>ModeActivationKind</code>	0..1	attr	Specifies if the event is raised on entering or exiting a specific mode or is raised on the transition between two modes.
mode (ordered)	<code>ModeDeclaration</code>	0..2	iref	The referenced mode or the transition between two modes raises this <code>SwcModeSwitchEvent</code> . <b>InstanceRef implemented by:</b> <code>RModelInAtomicSwcInstanceRef</code>

**Table 7.19: SwcModeSwitchEvent**

**[constr\_1946] Existence of attribute `SwcModeSwitchEvent.activation`** [For each `SwcModeSwitchEvent`, attribute `activation` shall exist **at the time when the RTE is generated.**]()

**[constr\_1947] Existence of reference `SwcModeSwitchEvent.mode`** [For each `SwcModeSwitchEvent`, the reference to `ModeDeclaration` in the role `mode` shall exist **at the time when the RTE is generated.**]()

<b>Enumeration</b>	<b>ModeActivationKind</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration
<b>Note</b>	Kind of mode switch condition used for activation of an event, as further described for each enumeration field.
<b>Literal</b>	<b>Description</b>
onEntry	On entering the referred mode. <b>Tags:</b> atp.EnumerationLiteralIndex=0
onExit	On exiting the referred mode. <b>Tags:</b> atp.EnumerationLiteralIndex=1
onTransition	On transition of the 1st referred mode to the 2nd referred mode. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 7.20: ModeActivationKind**

<b>Class</b>	<b>ModeSwitchedAckEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when the referenced ModeSwitchPoint has been processed or an error occurred.			
<b>Base</b>	ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, Multilanguage Referrable, RTEEvent, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
eventSource	ModeSwitchPoint	0..1	ref	The referenced ModeSwitchPoint raises this Mode SwitchedAckEvent when the ModeSwitchPoint has been processed.

**Table 7.21: ModeSwitchedAckEvent**

**[constr\_1948] Existence of attribute `ModeSwitchedAckEvent.eventSource`**  
 [For each `ModeSwitchedAckEvent`, attribute `eventSource` shall exist **at the time when the RTE is generated.**]()

<b>Class</b>	<b>ExternalTriggerOccurredEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when the referenced Trigger has occurred.			
<b>Base</b>	ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, Multilanguage Referrable, RTEEvent, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
trigger	Trigger	0..1	iref	The referenced Trigger raises this ExternalTrigger OccurredEvent. <b>InstanceRef implemented by:</b> RTriggerInAtomicSwc InstanceRef

**Table 7.22: ExternalTriggerOccurredEvent**

**[constr\_1949] Existence of attribute `ExternalTriggerOccurredEvent.trigger`**  
 [For each `ExternalTriggerOccurredEvent`, attribute `trigger` shall exist **at the time when the RTE is generated.**]()

<b>Class</b>	<b>InternalTriggerOccurredEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when the referenced InternalTriggeringPoint has occurred.			
<b>Base</b>	ARObject, <a href="#">AbstractEvent</a> , AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">RTEEvent</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
eventSource	<a href="#">InternalTriggeringPoint</a>	0..1	ref	The referenced InternalTriggeringPoint raises this InternalTriggerOccurredEvent.

**Table 7.23: InternalTriggerOccurredEvent**

**[constr\_1950] Existence of attribute [InternalTriggerOccurredEvent.eventSource](#)** [For each [InternalTriggerOccurredEvent](#), the attribute [eventSource](#) shall exist **at the time when the RTE is generated.**]()

<b>Class</b>	<b>InitEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">AbstractEvent</a> , AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">RTEEvent</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 7.24: InitEvent**

<b>Class</b>	<b>TransformerHardErrorEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This 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.			
<b>Base</b>	ARObject, <a href="#">AbstractEvent</a> , AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">RTEEvent</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
operation	<a href="#">ClientServerOperation</a>	0..1	iref	This represents the ClientServerOperation for which the transformer can raise this TransformerHardErrorEvent. <b>InstanceRef implemented by:</b> <a href="#">POperationInAtomicSwcInstanceRef</a>
requiredTrigger	<a href="#">Trigger</a>	0..1	iref	This represents the Trigger for which the transformer can raise this TransformerHardErrorEvent. <b>InstanceRef implemented by:</b> <a href="#">RTriggerInAtomicSwcInstanceRef</a>

**Table 7.25: TransformerHardErrorEvent**

**[constr\_1397] Existence of attributes of [TransformerHardErrorEvent](#)** [For any given [TransformerHardErrorEvent](#), **either** the attribute [TransformerHardErrorEvent.operation](#) **or** [TransformerHardErrorEvent.requiredTrigger](#) shall exist **at the time when the contract phase generation is executed.**]()

In other words, the attributes `operation` and `requiredTrigger` 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.

] (*RS\_SWCT\_03040*)

<b>Class</b>	<b>OsTaskExecutionEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SwComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This RTEEvent is supposed to execute RunnableEntities which have to react on the execution of specific OsTasks. Therefore, this event is unconditionally raised whenever the OsTask on which it is mapped is executed. The main use case for this event is scheduling of Runnables of Complex Drivers which have to react on task executions. <b>Tags:</b> atp.Status=draft			
<b>Base</b>	<i>ARObject</i> , <i>AbstractEvent</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>Multilanguage</i> , <i>Referrable</i> , <i>RTEEvent</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

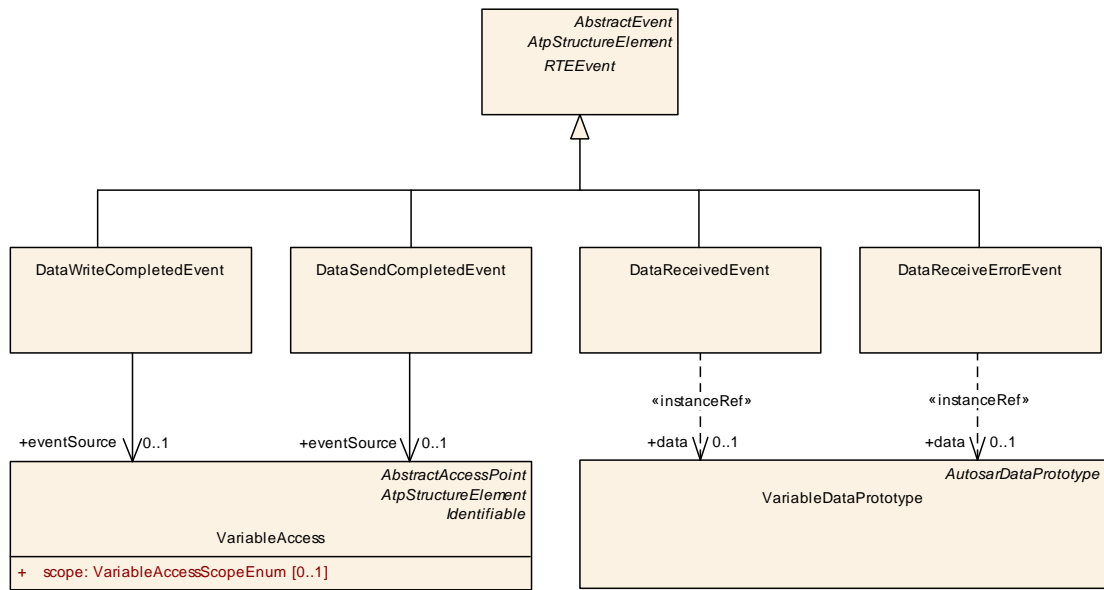
**Table 7.26: OsTaskExecutionEvent**

**[constr\_10016] Applicability of `OsTaskExecutionEvent`** [An `OsTaskExecutionEvent` is only applicable for a `SwcInternalBehavior` in the context of a `ComplexDeviceDriverSwComponentType`, `EcuAbstractionSwComponentType`, or `ServiceSwComponentType` **at any time in the workflow.**]()

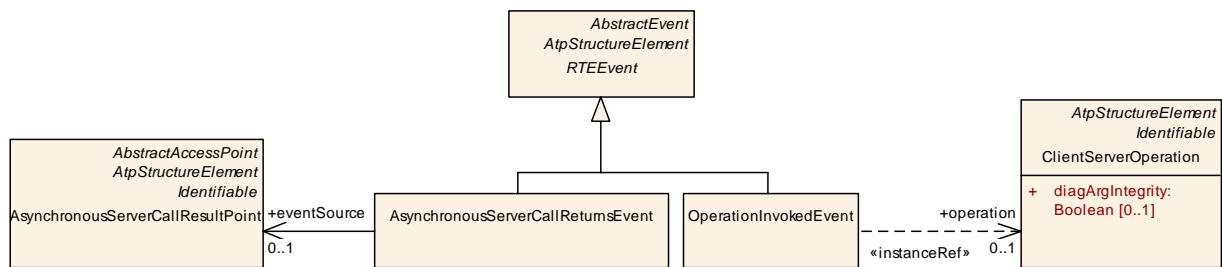
### 7.3.1 Defining an Event

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 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**



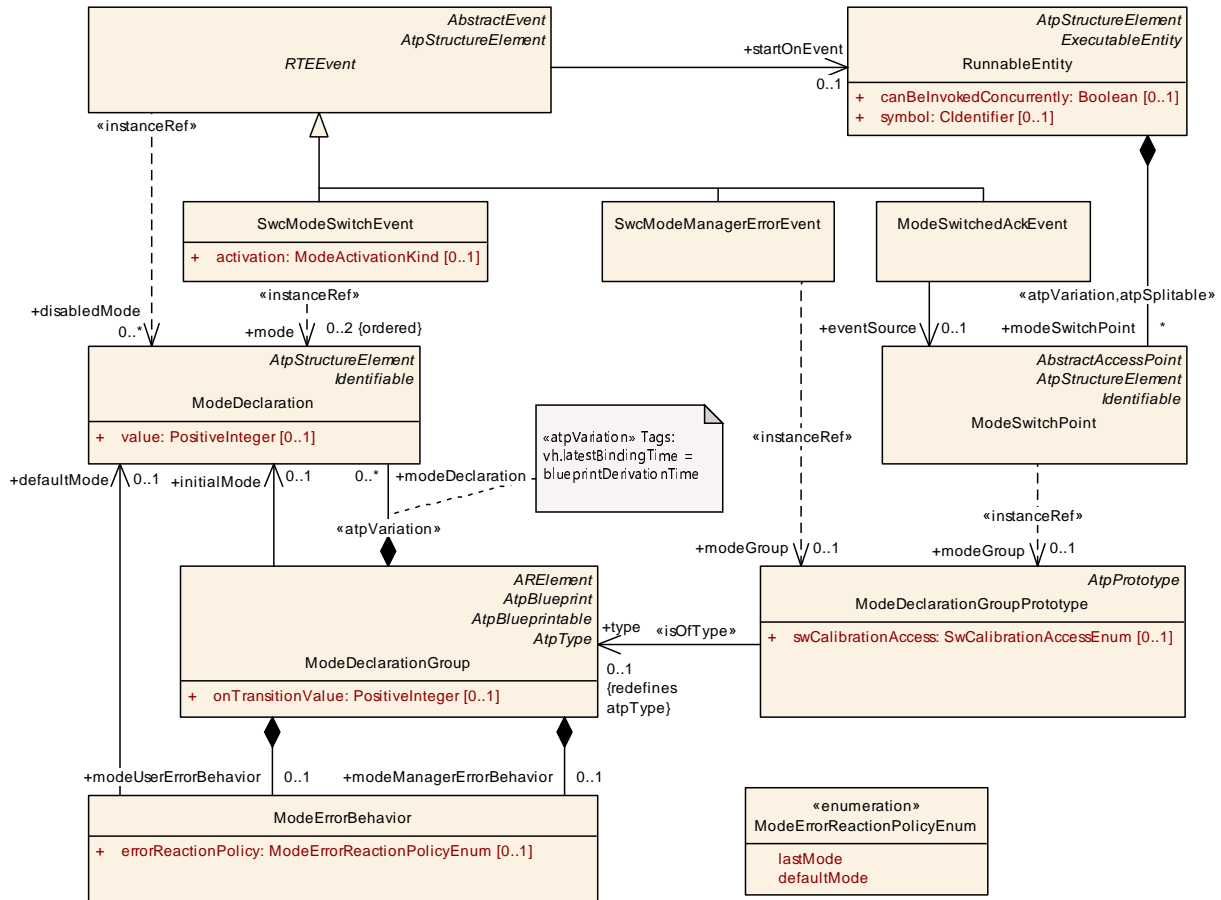


Figure 7.10: RTEEvents used in the context of mode communication

Please note that more explanation about the semantics of the meta-classes **SwcModeManagerErrorEvent** and **ModeErrorBehavior** can be found in section 9.4.

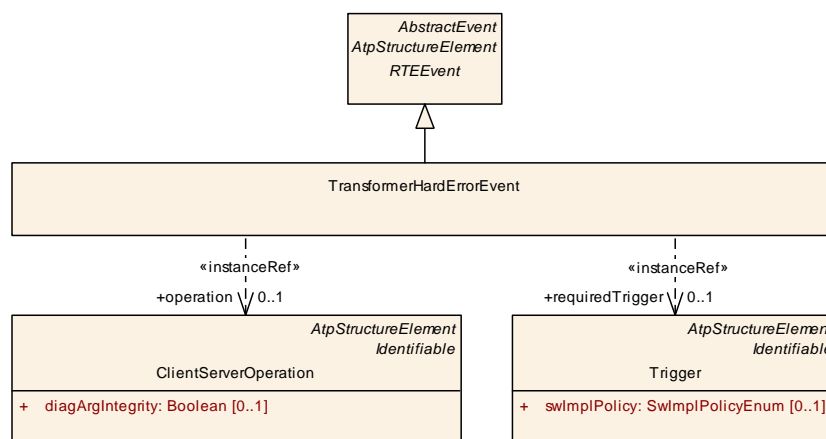
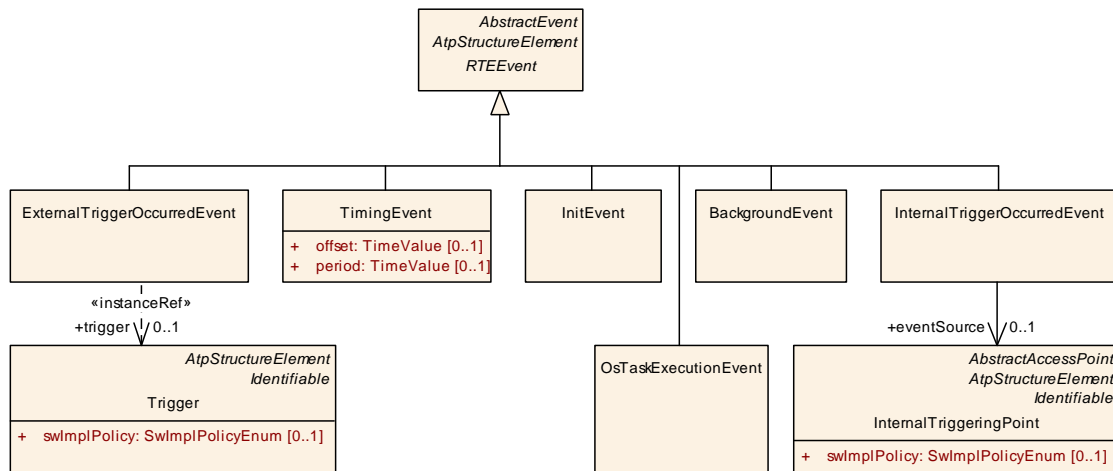


Figure 7.11: RTEEvent used in the context of data transformation



**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.

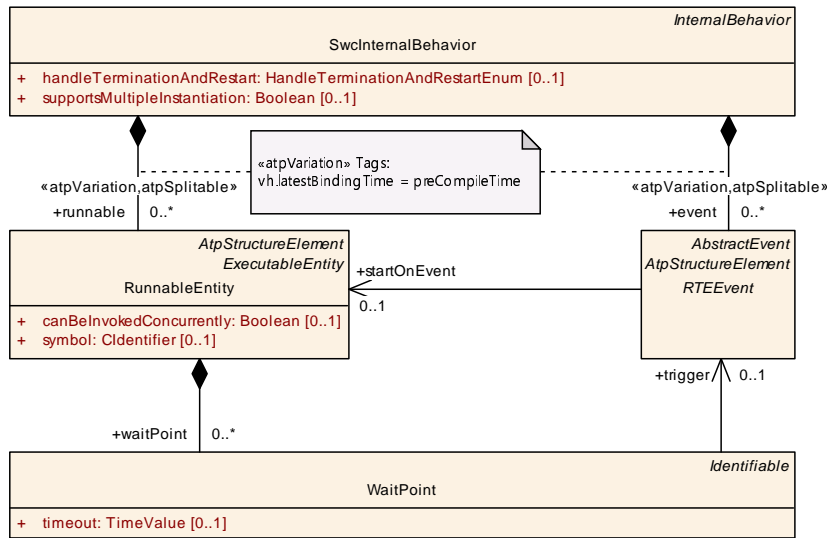
### 7.3.2 Defining how to Respond to an Event

**[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.] (RS\_SWCT\_03040)

**[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 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.] (RS\_SWCT\_03040)

Figure 7.13 gives an overview of the modeling of WaitPoint.



**Figure 7.13: Description of the interaction between an RTEEvent and RunnableEntities**

**[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 time<sup>3</sup>.]()

**[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 RunnableEntities** [It is in general possible that a single RTEEvent can be used to trigger WaitPoints in different RunnableEntities.]()

Concerning DataReceivedEvents consider as well [constr\_2021].

<b>Class</b>	WaitPoint			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This defines a wait-point for which the RunnableEntity can wait.			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>



<sup>3</sup>This constraint is valid at least in the ISO 17356-3 [32] standard where an extended task (that can have wait points) can only exist a single time in the context of the scheduler.



Class	WaitPoint			
timeout	TimeValue	0..1	attr	Time in seconds before the WaitPoint times out and the blocking wait call returns with an error indicating the timeout.
trigger	RTEEvent	0..1	ref	This is the RTEEvent this WaitPoint is waiting for.

**Table 7.27: WaitPoint**

**[constr\_1951] Existence of attribute `WaitPoint.timeout`** [For each `WaitPoint`, attribute `timeout` shall exist **at the time when the RTE is generated.**]()

**[constr\_1952] Existence of reference `WaitPoint.trigger`** [For each `WaitPoint`, the reference to `RTEEvent` in the role `trigger` shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1096] `SwcModeSwitchEvent` and `WaitPoint`** [A `RunnableEntity` that has a `WaitPoint` shall not be referenced by a `SwcModeSwitchEvent` **at the time when the contract phase generation is executed.**]()

**[TPS\_SWCT\_01320] `RunnableEntity`s of category 2** [`RunnableEntity`s 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.] (*RS\_SWCT\_03040*)

**[constr\_1097] `RunnableEntity` that has a `WaitPoint`** [A `RunnableEntity` that has a `WaitPoint` shall not be referenced by an `RTEEvent` that has a reference in the role `disabledMode` **at the time when the RTE is generated.**]()

**[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.] (*RS\_SWCT\_03040*)

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 `RunnableEntity`s within a specific `Atomic-SwComponentType` will need to communicate among each other.

**[TPS\_SWCT\_01321] Communication among `RunnableEntity`s** [The RTE needs to provide synchronization mechanisms to the `RunnableEntity`s 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 RunnableEntities of different instances of the same AtomicSwComponentType is only supported via PortPrototypes.](RS\_SWCT\_00120)

Several concepts for implementing communication among RunnableEntities 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 RunnableEntities within one AtomicSwComponentType.

Two possible approaches for formal specification of this kind of communication are described:

- Specifying that several RunnableEntities belong in a specific ExclusiveArea
- Specifying the data exchanged between the RunnableEntities

### 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).

<b>Class</b>	<b>ExclusiveArea</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
<b>Note</b>	Prevents an executable entity running in the area from being preempted.			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

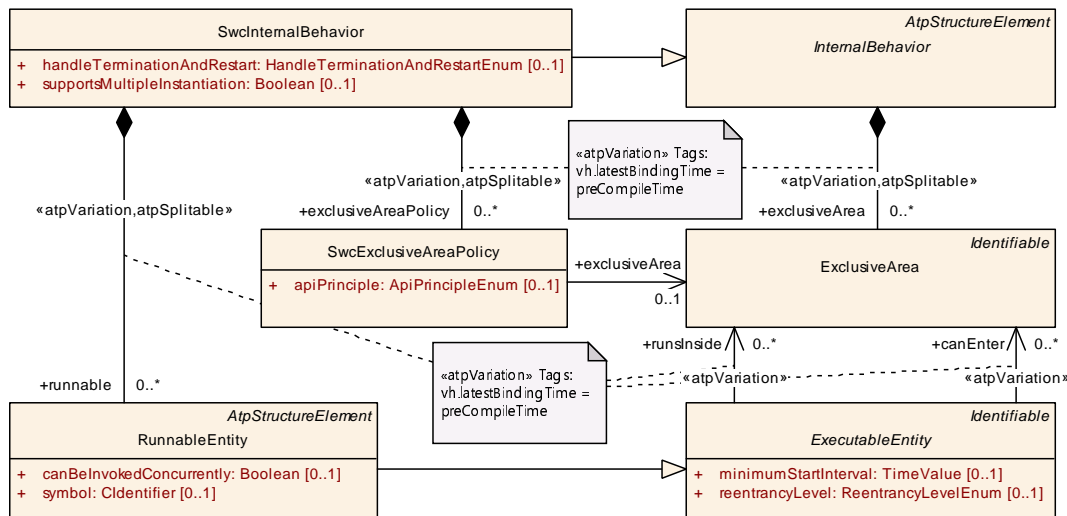
**Table 7.28: ExclusiveArea**

[TPS\_SWCT\_01031] **ExclusiveArea** [An ExclusiveArea merely specifies a constraint on the scheduling policy and configuration of the RTE:

If two or more RunnableEntities refer to the same ExclusiveArea only one of these RunnableEntities is allowed to be executed while being inside that ExclusiveArea.](RS\_SWCT\_00120, RS\_SWCT\_02090)

In other words: these RunnableEntities shall not run concurrently (preempt each other) while executing inside the ExclusiveArea.

Please find more details about the formal definition of meta-class ExclusiveArea in Figure 7.14.

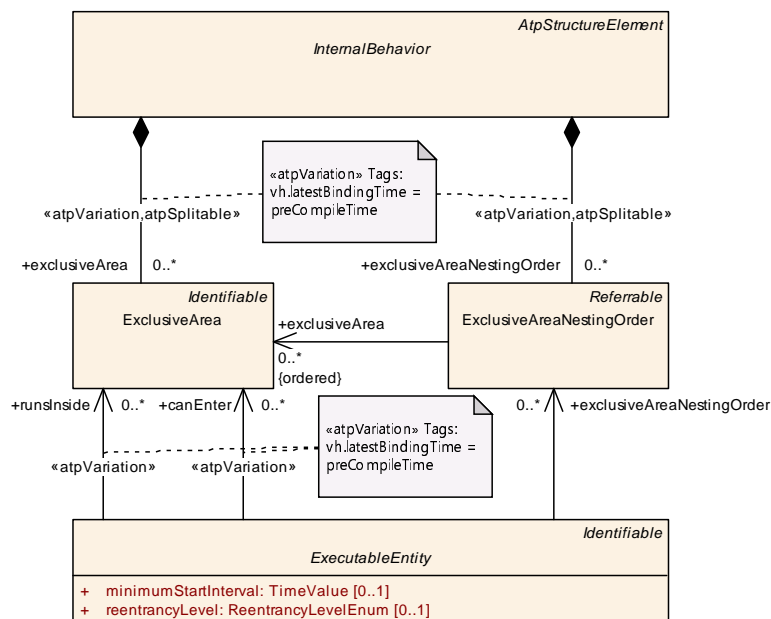


**Figure 7.14: Description of logical exclusive areas**

[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.canEnter** shall exist).

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.runsInside**.) (*RS\_SWCT\_00120, RS\_SWCT\_02090*)

Please note that the options for entering an **ExclusiveArea** are documented in section 7.4.1.1 and section 7.4.1.2



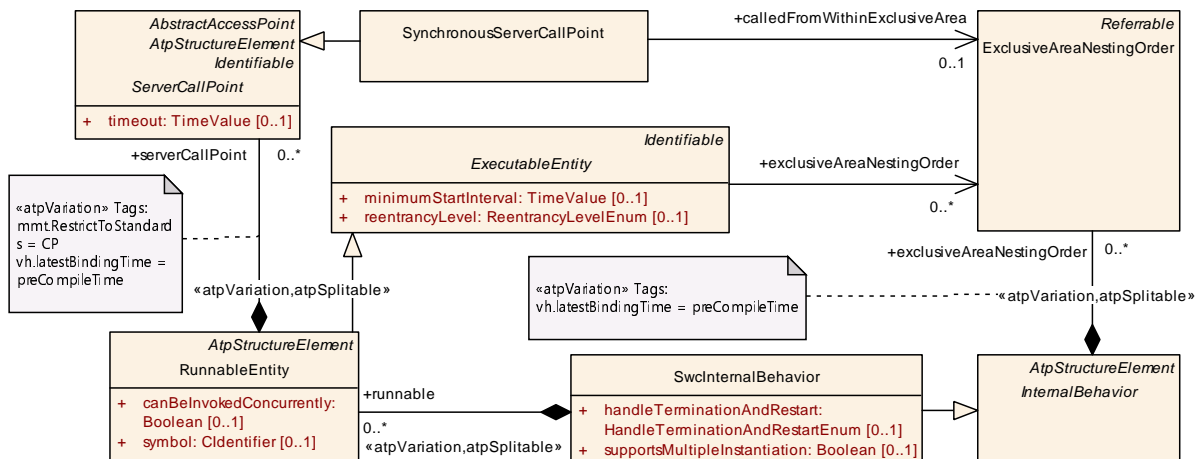
**Figure 7.15: Description of nested usage of **ExclusiveArea****

[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*)

An **ExclusiveAreaNestingOrder** is aggregated by the **InternalBehavior** that in turn also owns **RunnableEntity**.



**Figure 7.16: Nested usage of ExclusiveArea and the impact on SynchronousServerCallPoint**

<b>Class</b>	<b>ExclusiveAreaNestingOrder</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
<b>Note</b>	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.			
<b>Base</b>	ARObject, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
exclusiveArea (ordered)	ExclusiveArea	*	ref	This represents a specific scenario of how Exclusive Areas can be used in terms of the nesting order.

**Table 7.29: ExclusiveAreaNestingOrder**

[TPS\_SWCT\_01460] Relation of **SynchronousServerCallPoint** to **ExclusiveAreaNestingOrder** [In case other **RunnableEntities** 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.

#### 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 [RunnableEntities](#) 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 [RunnableEntities](#) '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 [RunnableEntities](#) 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, [RunnableEntities](#) '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 [RunnableEntities](#) 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' (or 'r2') in the context of 't1' (or '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 [6] the concept can be applied to software-components as well.

### 7.4.1.3 Configuration of API Generation

For certain usage scenarios of `ExclusiveAreas` it is considered advantageous if each `RunnableEntity` uses a distinct set of enter and exit APIs.

This distinct set of APIs support `ExclusiveArea` implementations where for the `RunnableEntity(s)` with the highest priority the lock is omitted.

This is possible when the `RunnableEntity(s)` with the highest priority can't be interrupted by `RunnableEntitys` scheduled with lower priority.

To support this kind of implementation, the software-component description has to state (by means of attribute `SwcInternalBehavior.exclusiveAreaPolicy.apiPrinciple`) that it requests APIs individually for each `RunnableEntity` referencing an `ExclusiveArea` in the role `canEnter`.

<b>Class</b>	<b>SwcExclusiveAreaPolicy</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
<b>Note</b>	Options how to generate the ExclusiveArea related APIs. If no SwcExclusiveAreaPolicy is specified for an ExclusiveArea the default values apply.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
apiPrinciple	ApiPrincipleEnum	0..1	attr	Specifies for this ExclusiveArea if either one common set of Enter and Exit APIs for the whole software component is requested from the Rte or if the set of Enter and Exit APIs is expected per RunnableEntity. The default value is "common".
exclusiveArea	ExclusiveArea	0..1	ref	This reference represents the ExclusiveArea for which the policy applies.

**Table 7.30: SwcExclusiveAreaPolicy**

**[constr\_1953] Existence of attribute `SwcExclusiveAreaPolicy.apiPrinciple`**  
 [For each `SwcExclusiveAreaPolicy` that refers to an `exclusiveArea`, attribute `apiPrinciple` shall exist **at the time when the RTE is generated.**]()

<b>Enumeration</b>	<b>ApiPrincipleEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::InternalBehavior
<b>Note</b>	This enumeration represents the ability to control the granularity of API generation.
<b>Literal</b>	<b>Description</b>
common	The Rte or SchM API is provided for the whole software component / BSW Module <b>Tags:</b> atp.EnumerationLiteralIndex=0





Enumeration	ApiPrincipleEnum
perExecutable	The Rte or SchM API is provided for a specific ExecutableEntity of a software component / BSW Module <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 7.31: ApiPrincipleEnum**

**[TPS\_SWCT\_01713] ExclusiveArea is entered and exited by a common set of APIs** [If the value of attribute `SwcExclusiveAreaPolicy.apiPrinciple` is set to `ApiPrincipleEnum.common` then the RTE provides **one set** of enter and exit APIs to be shared among all `RunnableEntity`s of the whole software-component.] ([RS\\_SWCT\\_00120](#), [RS\\_SWCT\\_02090](#))

In this case, the same enter and exit code is executed by all affected `RunnableEntity`s and there is no way to have a special treatment for the `RunnableEntity`(s) executed in the context with the highest priority.

**[TPS\_SWCT\_01714] ExclusiveArea is entered and exited by an individual set of APIs** [If the value of attribute `SwcExclusiveAreaPolicy.apiPrinciple` is set to `ApiPrincipleEnum.perExecutable` then the RTE provides **individual** sets of APIs for entering and exiting `ExclusiveAreas` for each affected `RunnableEntity`.] ([RS\\_SWCT\\_00120](#), [RS\\_SWCT\\_02090](#))

In this case, the implementation of enter and exit code for the `RunnableEntity` executed in the execution context with the highest priority can be left empty.

In order to avoid the existence of contradicting settings of `SwcExclusiveAreaPolicies` for one `ExclusiveArea` [[constr\\_1468](#)] applies.

**[constr\_1468] Limitation on the number of SwcExclusiveAreaPolicies** [An `ExclusiveArea` shall only be referenced by **at most one** `SwcExclusiveAreaPolicy` **at the time when the contract phase generation is executed.**]()

## 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](#))

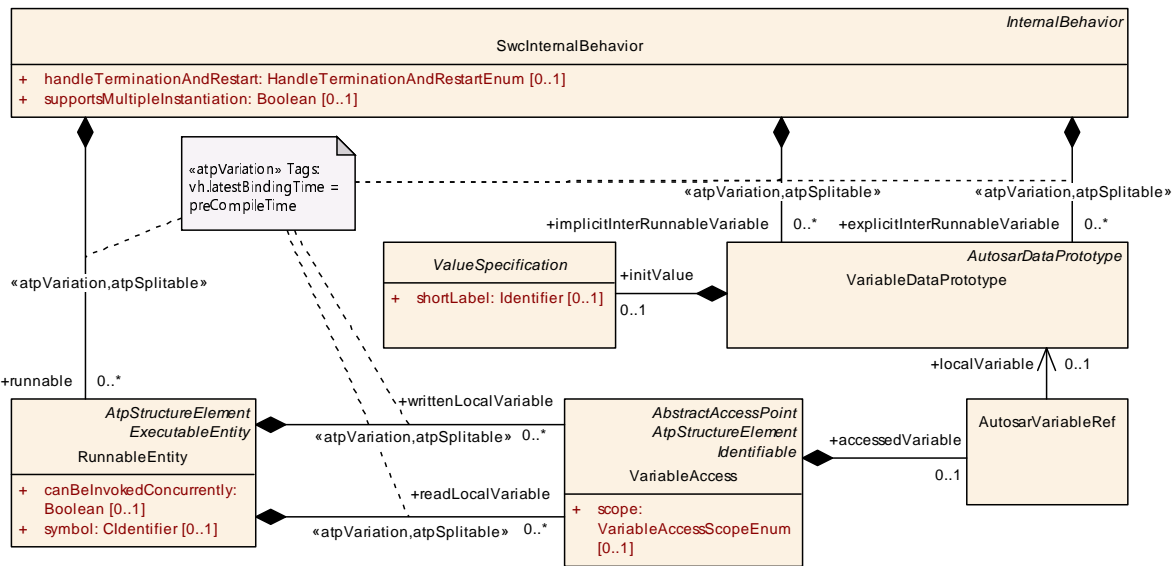
**[TPS\_SWCT\_01053] Relationship of interchanged data with RunnableEntity**  
 [Furthermore, the relationship of these data with RunnableEntity shall be specified.

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 AutosarVariableRef.localVariable to the respective VariableDataPrototype exists.](RS\_SWCT\_00120, RS\_SWCT\_02090)

**[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.](RS\_SWCT\_03040)



**Figure 7.17: implicitInterRunnableVariable vs. explicitInterRunnableVariable**

**[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.

This rule shall be imposed at the time when the contract phase generation is executed.](

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`.]([RS\\_SWCT\\_03040](#))

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 a `VariableDataPrototype` used in the role `explicitInterRunnableVariable`.

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`.]([RS\\_SWCT\\_00120](#), [RS\\_SWCT\\_02090](#))

This approach is in detail described in chapter [2.3.1.4](#).

[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:

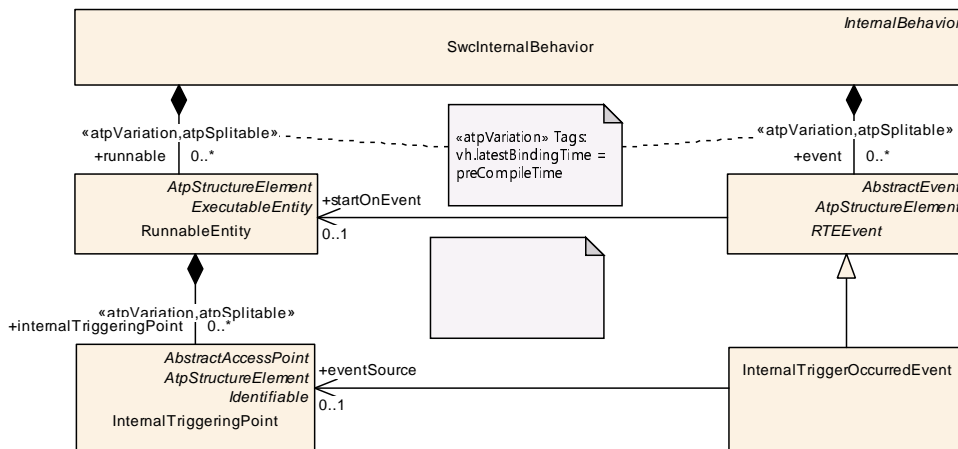
- ImplementationDataType of category DATA\_REFERENCE, or
- ImplementationDataType that contains subElements that (after potential indirections via TYPE\_REFERENCE are resolved) are of category DATA\_REFERENCE.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

### 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.



**Figure 7.18: Model of software-component Inter Runnable Triggering**

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.

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 `InternalTriggerOccurredEvent` which references the particular `InternalTriggeringPoint` and additionally the to-be-activated `RunnableEntity`.] (*RS\_SWCT\_03040*)

**[TPS\_SWCT\_01022] Queued processing of internal trigger** [The attribute `InternalTriggeringPoint.swImplPolicy` can be used to specify a requirement whether the internal triggering of the enclosing `RunnableEntity` using the given `InternalTriggeringPoint` shall be queued.] (*RS\_SWCT\_03040*)

**[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).

This rule shall be imposed **at the time when the RTE is generated.**] ()

<b>Class</b>	<code>InternalTriggeringPoint</code>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Trigger			
<b>Note</b>	If a <code>RunnableEntity</code> owns an <code>InternalTriggeringPoint</code> it is entitled to trigger the execution of <code>RunnableEntities</code> of the corresponding software-component.			
<b>Base</b>	<code>ARObject</code> , <code>AbstractAccessPoint</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
<code>swImplPolicy</code>	<code>SwImplPolicyEnum</code>	0..1	attr	This attribute, when set to value <code>queued</code> , allows for a queued processing of Triggers.

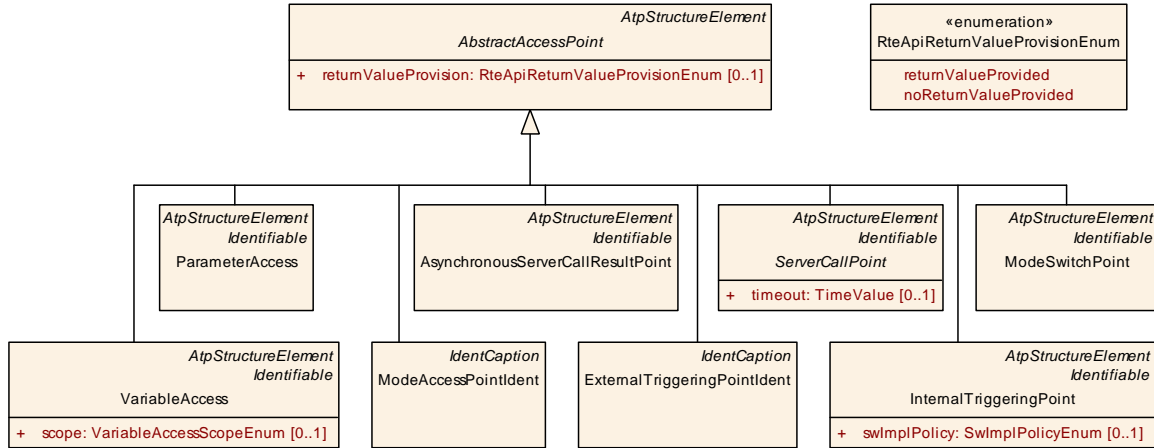
**Table 7.32: InternalTriggeringPoint**

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.



**Figure 7.19: Modeling of the `AbstractAccessPoint`**

Access of `RunnableEntity`s to the different elements in `PortInterfaces` or the `InternalBehavior` are modeled by a set of meta-classes specific to the communication pattern and the kind of access.

Nevertheless, all of those meta-classes inherit from `AbstractAccessPoint` in order to enable the ability to be referenced in a harmonized way to by additional descriptions.

<b>Class</b>	<b>AbstractAccessPoint</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::AccessCount			
<b>Note</b>	Abstract class indicating an access point from an ExecutableEntity.			
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
<b>Subclasses</b>	AsynchronousServerCallResultPoint, ExternalTriggeringPointIdent, InternalTriggeringPoint, ModeAccessPointIdent, ModeSwitchPoint, ParameterAccess, ServerCallPoint, VariableAccess			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
returnValueProvision	RteApiReturnValueProvisionEnum	0..1	attr	This attribute controls the provision of return values for RTE APIs that correspond to the enclosing access point.

**Table 7.33: AbstractAccessPoint**

<b>Enumeration</b>	<b>RteApiReturnValueProvisionEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::AccessCount
<b>Note</b>	This meta-class provides values to control how return values from RTE APIs are provided.
<b>Literal</b>	<b>Description</b>
noReturnValueProvided	The RTE API shall not provide a return value. <b>Tags:</b> atp.EnumerationLiteralIndex=1
returnValueProvided	The RTE API shall provide a return value. <b>Tags:</b> atp.EnumerationLiteralIndex=0

**Table 7.34: RteApiReturnValueProvisionEnum**

`RunnableEntity` that access `DataPrototypes` in the context of `PortPrototypes` are **not** allowed to extend the data access to sub-elements of the respective `DataPrototypes`.

For example, assume a `DataPrototype` that effectively implements a structure of two elements A and B. It is **not** supported to only send or receive only element A or B of the structure.

This assertion leads to the existence of `[constr_1429]` and, by extension, `[constr_1430]`.

**[constr\_1429] Access to data within `PortPrototypes` from within `RunnableEntities`** [For a `VariableAccess` that is aggregated in the roles

- `RunnableEntity.dataWriteAccess`
- `RunnableEntity.dataReadAccess`
- `RunnableEntity.dataSendPoint`
- `RunnableEntity.dataReceivePointByArgument`
- `RunnableEntity.dataReceivePointByValue`

the existence of the following attributes is not allowed:

- `VariableAccess.accessedVariable.autosarVariable.contextDataPrototype`
- `VariableAccess.accessedVariable.autosarVariable.rootVariableDataPrototype`
- `VariableAccess.accessedVariable.autosarVariableInImplDatatype`
- `VariableAccess.accessedVariable.localVariable`

In other words: in this case, only the references

- `VariableAccess.accessedVariable.autosarVariable.portPrototype` and
- `VariableAccess.accessedVariable.autosarVariable.targetDataPrototype`

shall exist and the latter shall **exclusively** refer to a `VariableDataPrototype` that is aggregated as either

- `SenderReceiverInterface.dataElement` or
- `NvDataInterface.nvData`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1430] Access to local data from within `RunnableEntities`** [For `VariableAccess` that is aggregated in the roles



- `RunnableEntity.writtenLocalVariable`
- `RunnableEntity.readLocalVariable`

the existence of the following attributes is not allowed:

- `VariableAccess.accessedVariable.autosarVariableInImpl-Datatype`
- `VariableAccess.accessedVariable.autosarVariable`

In other words, **only** the reference `VariableAccess.accessedVariable.localVariable` shall be used in this case.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1431] Access to parameters from within `RunnableEntity`s** [For a `ParameterAccess` that is aggregated in the role `RunnableEntity.parameterAccess` the existence of the following attributes is not allowed:

- `ParameterAccess.accessedParameter.autosarParameter.contextDataPrototype`
- `ParameterAccess.accessedParameter.autosarParameter.rootParameterDataPrototype`

In other words: in this case, **one** of the following alternatives is allowed to exist:

- a combination of
  - `ParameterAccess.accessedParameter.autosarParameter.portPrototype` and
  - `ParameterAccess.accessedParameter.autosarParameter.targetDataPrototype` that **exclusively** refers to a `ParameterDataPrototype` aggregated by a `ParameterInterface` in the role `parameter`.
- `ParameterAccess.accessedParameter.localParameter` that refers to a `ParameterDataPrototype` that is either aggregated as
  - `InternalBehavior.constantMemory` or
  - `SwcInternalBehavior.perInstanceParameter` or
  - `SwcInternalBehavior.sharedParameter`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

### 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](#)).

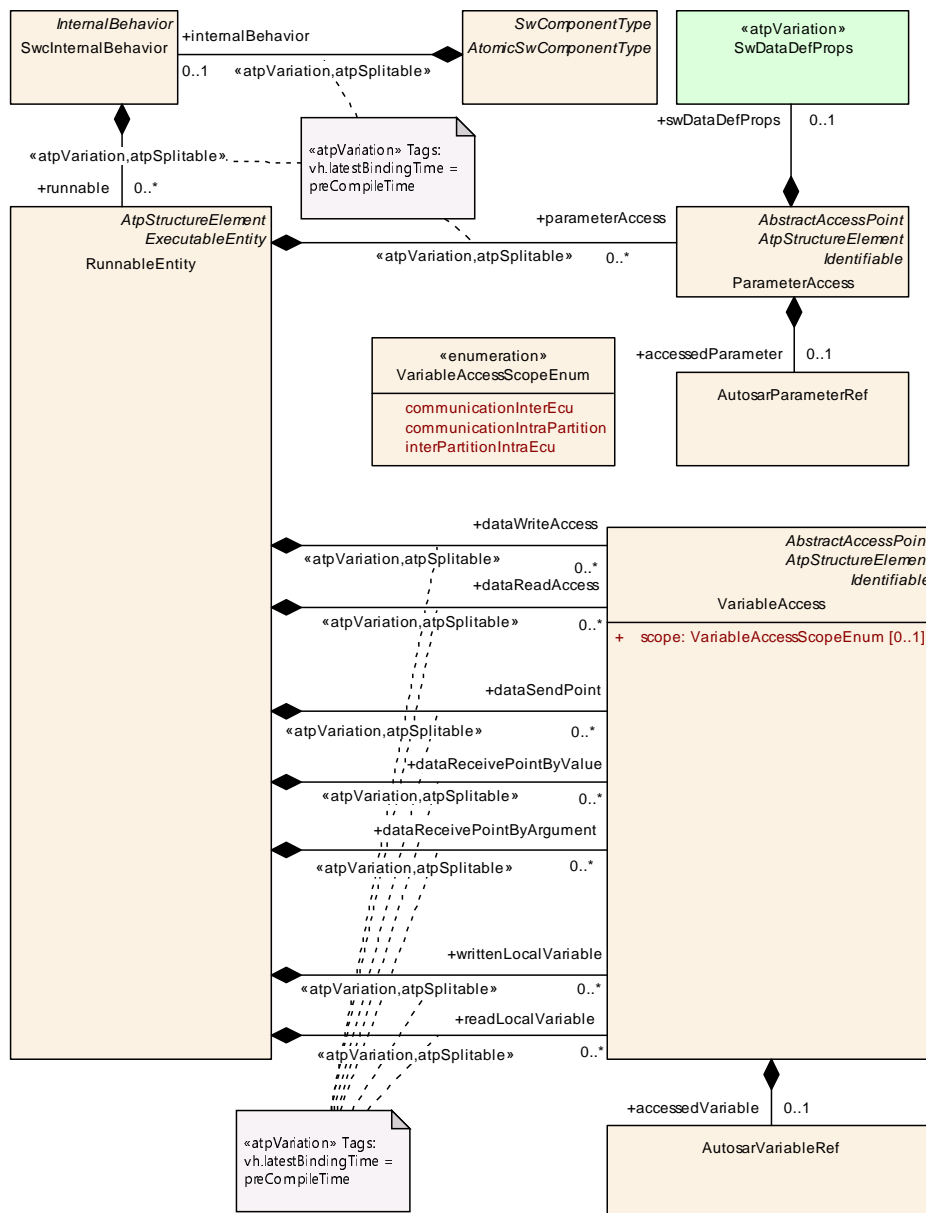
The motivation for the differentiation between “implicit” and “explicit” was originally the characteristics of the RTE specification that foresaw an API for handling a [dataSendPoint](#) or [dataReceivePointByValue](#) in contrast to the *data-access* that was supposed to be part of the function signature (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*)



**Figure 7.20: DataReadAccess and DataWriteAccess**

**[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.

<b>Class</b>	<b>VariableAccess</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
<b>Note</b>	The presence of a <code>VariableAccess</code> implies that a <code>RunnableEntity</code> needs access to a <code>VariableDataPrototype</code> . The kind of access is specified by the role in which the class is used.			
<b>Base</b>	<i>ARObject</i> , <i>AbstractAccessPoint</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
accessed Variable	<code>AutosarVariableRef</code>	0..1	aggr	This denotes the accessed variable.
scope	<code>VariableAccessScopeEnum</code>	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 7.35: VariableAccess**

**[constr\_1954] Existence of attribute `VariableAccess.accessedVariable`** [For each `VariableAccess`, attribute `accessedVariable` shall exist **at the time when the contract phase generation is executed.**] ()

<b>Enumeration</b>	<b>VariableAccessScopeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements
<b>Note</b>	This enumeration defines scopes for communication.
<b>Literal</b>	<b>Description</b>
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. <b>Tags:</b> atp.EnumerationLiteralIndex=0





Enumeration	VariableAccessScopeEnum
communicationIntraPartition	This case is foreseen to express that the corresponding communication shall <b>not</b> cross the boundary of a partition. <b>Tags:</b> atp.EnumerationLiteralIndex=1
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. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 7.36: 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`

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

This aspect is depicted in Figure 7.20.

### 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` **at the time when the contract phase generation is executed.**]()

**[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 only made available after the `RunnableEntity` terminates.
- The presence of a `dataSendPoint` per definition lets the corresponding `RunnableEntity` attain cat. 1B.

] (*RS\_SWCT\_00200*)

**[constr\_1773] Value of attribute `dataSendPoint.returnValueProvision`** [All `RunnableEntity.dataSendPoint` that refer to the same `accessedVariable` shall define the identical value for attribute `returnValueProvision` **at the time when the contract phase generation is executed.**]()

Rationale for the existence of [constr\_1773]: different `RunnableEntity`s could aggregate `VariableAccess` in the role `dataSendPoint` with a different configuration of attribute `returnValueProvision`.

However, in such a case it would not be possible to generate the corresponding RTE API because the API exists once per software-component and it is therefore indispensable to have all affected `VariableAccess` aggregated in the role `dataSendPoint` agree on the configuration of attribute `returnValueProvision`.

This relation is sketched in Figure 7.21.

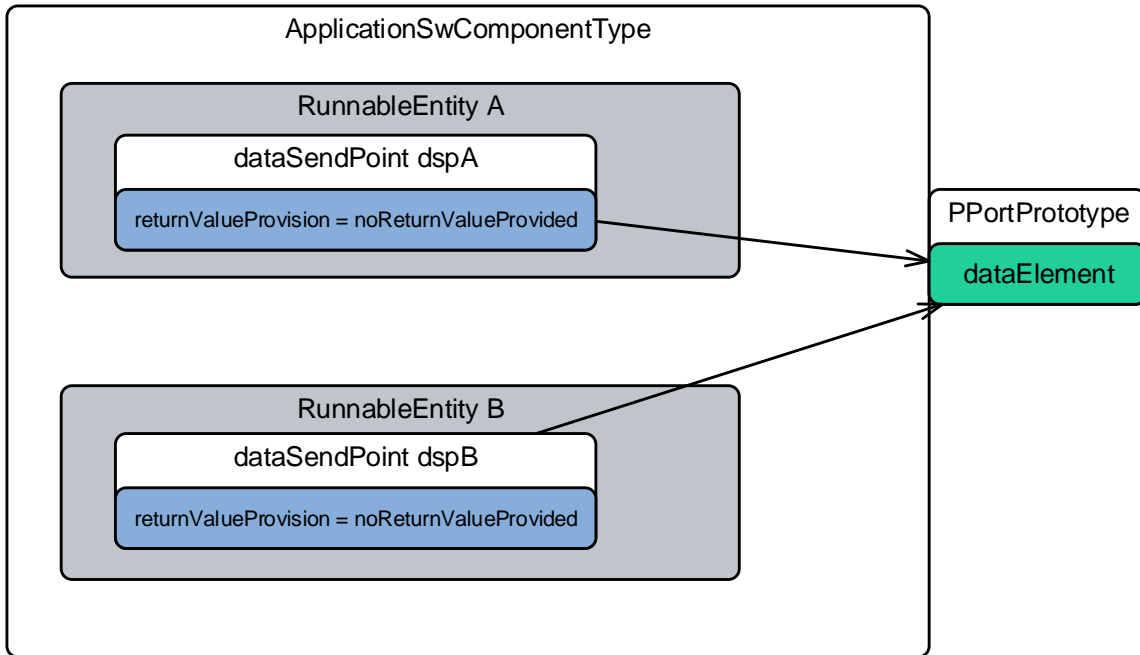


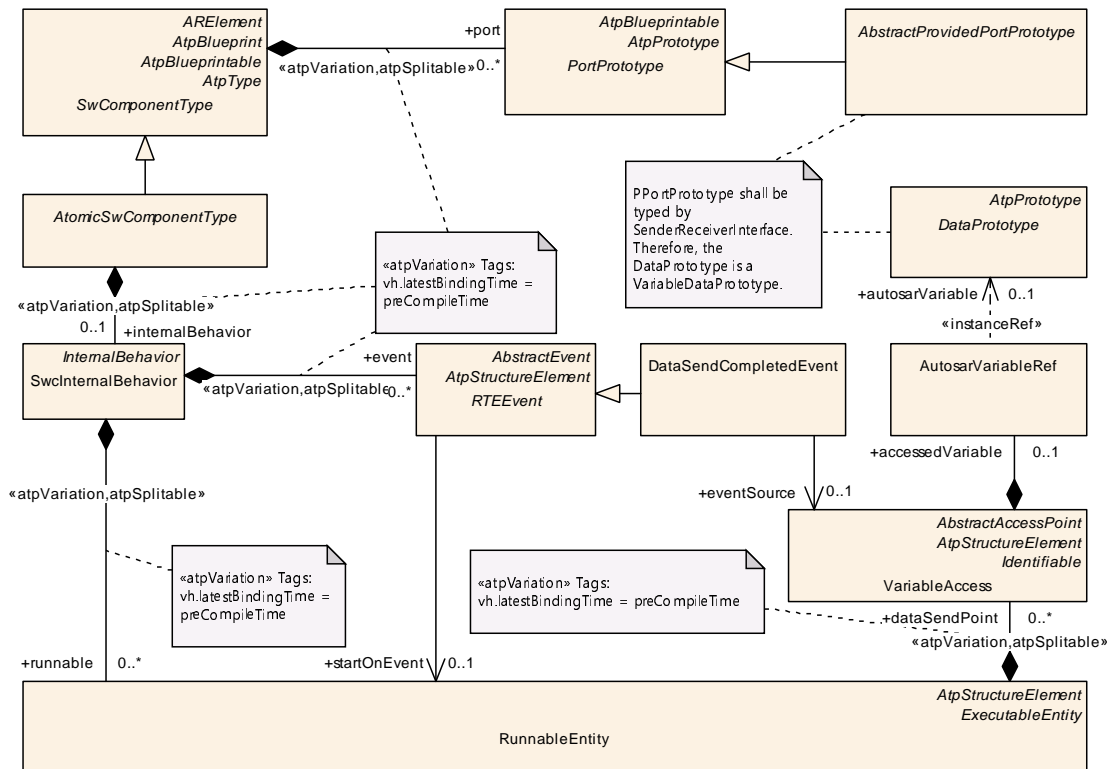
Figure 7.21: Modeling of attribute `dataSendPoint.returnValueProvision`

[TPS\_SWCT\_01663] `dataReadAccess` vs. `dataReceivePointByValue` or `dataReceivePointByArgument` [As opposed to the `dataReadAccess`:

- Using the `dataReceivePointByValue` or `dataReceivePointByArgument`, the `RunnableEntity` always “receives” the latest value of the `dataElement` for each call to the respective API during the execution of the `RunnableEntity`.
- When using a `dataReadAccess`, the value of the respective `dataElement` is received before the `RunnableEntity` starts and does not change during the execution of the `RunnableEntity` independently of the number of API calls for implicit reception.
- The presence of a `dataReceivePointByValue` or `dataReceivePointByArgument` per definition lets the corresponding `RunnableEntity` attain cat. 1B.

](RS\_SWCT\_00200)

For more details, please refer to section 4.9.



**Figure 7.22: 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.]()

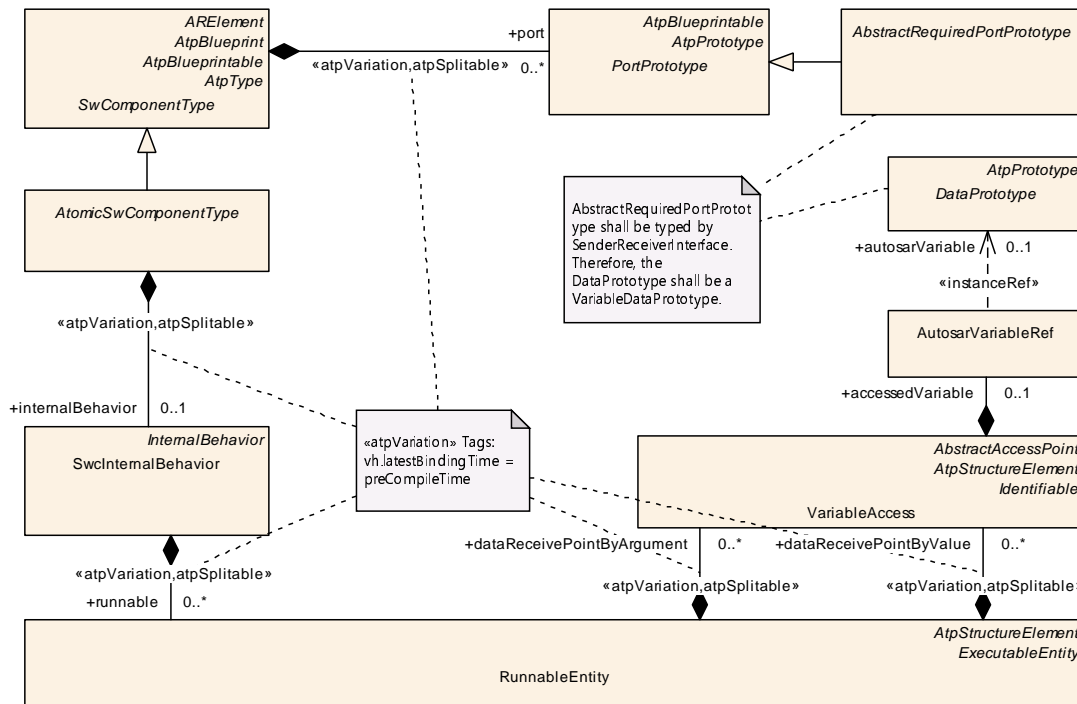
This aspect is visualized in Figure 7.23.

**[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 at the time when the contract phase generation is executed.**]()

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 way<sup>4</sup> 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.

<sup>4</sup>That 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 7.23: 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 `RunnableEntity`s may also have `dataReceivePointByValues/dataReceivePointByArguments` and consequently become `RunnableEntity`s of category 1B] (*RS\_SWCT\_00200*)

Please note that the categories of `RunnableEntity` are explained in section 7.2.4.4.

Similar to the `dataReadAccess`, constraints apply to the reference target of the `AutosarVariableRef` of `VariableAccess` in role `dataReceivePointByValue` or `dataReceivePointByArgument`.

**[constr\_1774] Value of attribute dataReceivePointByArgument.returnValueProvision** [All `RunnableEntity.dataReceivePointByArgument` that refer to the same `accessedVariable` shall define the identical value for attribute `returnValueProvision` at the time when the contract phase generation is executed.] ()

Rationale for the existence of [constr\_1774]: different `RunnableEntity`s could aggregate `VariableAccess` in the role `dataReceivePointByArgument` with a different configuration of attribute `returnValueProvision`.

However, in such a case it would not be possible to generate the corresponding RTE API because the API exists once per software-component and it is therefore indispensable to have all affected `VariableAccess` aggregated in the role `dataReceivePointByArgument` agree on the configuration of attribute `returnValueProvision`.

This relation is exemplarily (for the case of `dataSendPoint.returnValueProvision`) sketched in Figure 7.21.

**[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` **at the time when the contract phase generation is executed.**]()

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

Please note however, that in this case (in response to the presence of a `WaitPoint`) the `RunnableEntity` becomes category 2.

#### 7.5.1.4 Implicit Sending and Receiving

Implicit sending and receiving aims at the optimization of computation effort for sender-receiver communication.

Instead of executing the full amount of functionality for each call to a send-API or receive-API the implicit communication only receives implicitly received values latest

before the start of the execution of a `RunnableEntity` and sends implicitly sent values earliest after termination of the `RunnableEntity`.

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed**.] ()

**[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`.

This rule shall be imposed **at the time when the contract phase generation is executed**.] ()

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 at the time when the contract phase generation is executed**.] ()

**[constr\_1256] Acknowledgement feedback in n:1 writer case** [Within the scope of one `SwcInternalBehavior`, it is **not** allowed that two or more aggregated `RunnableEntities` 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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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.

The difference between implicit and explicit sender/receiver communication is explained in [TPS\_SWCT\_01331] and [TPS\_SWCT\_01663].

#### 7.5.1.5 DataSendCompletedEvent

[TPS\_SWCT\_01336] `dataSendPoint` also allows for the definition of a `DataSendCompletedEvent` [The `dataSendPoint` also allows for the definition of a `DataSendCompletedEvent`. 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 4.36.

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.

[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` **at the time when the RTE is generated.**]()

#### 7.5.1.6 DataWriteCompletedEvent

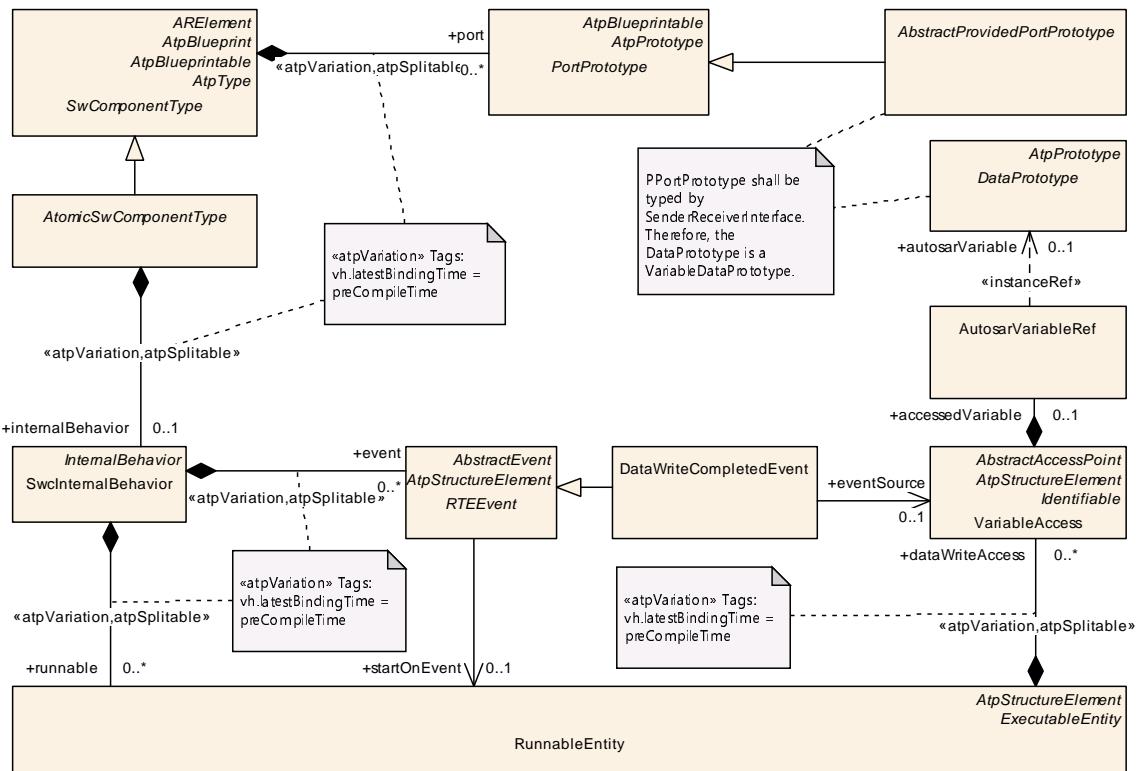
[TPS\_SWCT\_01557] `dataWriteAccess` also allows for the definition of a `DataWriteCompletedEvent` [The `dataWriteAccess` also allows for the definition of a `DataWriteCompletedEvent`. This `RTEEvent` occurs when the data has been successfully sent or when an error has occurred during sending.] (*RS\_SWCT\_00200*)

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 `PortPrototype` or `PortPrototype`, as is shown in Figure 4.36.

**[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*)

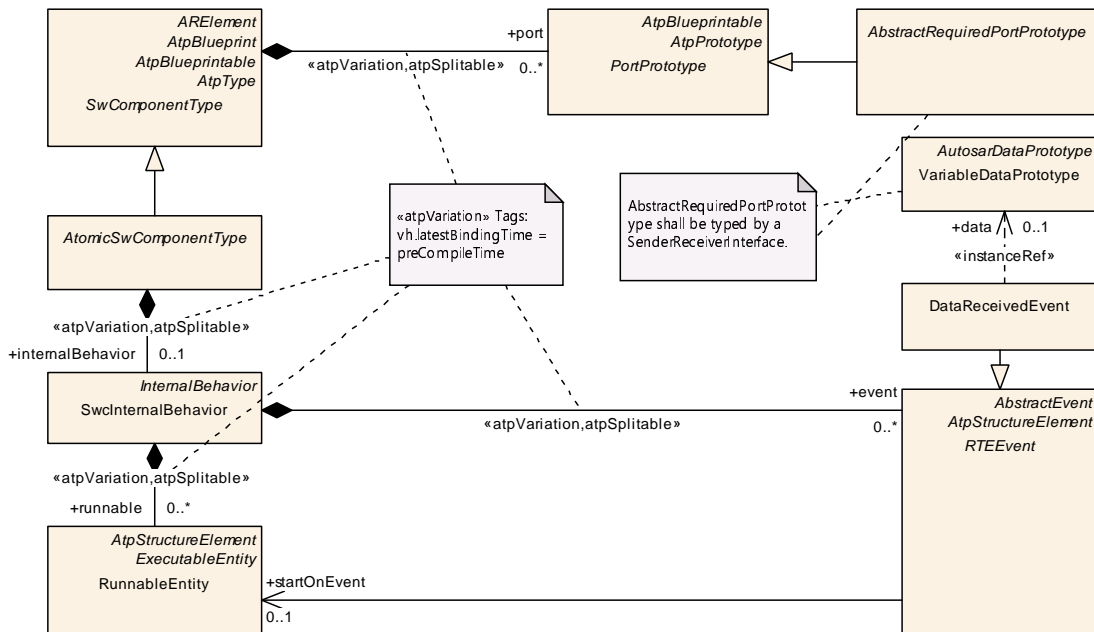
However, it is possible to configure the RTE such that when data is sent, the RTE will try to obtain the specified acknowledgement; possibly by waiting a certain `timeout` period.



**Figure 7.24: DataWriteAccess**

### 7.5.1.7 DataReceivedEvent

**[TPS\_SWCT\_01337] `DataReceivedEvent`** [A receiver is notified through the same event mechanism when a `VariableDataPrototype` is received. The `DataReceivedEvent` is directly associated with the corresponding `VariableDataPrototype`.] (*RS\_SWCT\_00200*)



**Figure 7.25: Receiver is notified by an event when new data has arrived**

### 7.5.1.8 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<sup>5</sup> 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)

<sup>5</sup>In 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.

**[constr\_10073] Existence of DataReceiveErrorEvent** [A `DataReceiveErrorEvent` shall only exist if it latest **at the time when the contract phase generation is executed** refers to a given `VariableDataPrototype` in the role `data` where either

- the `VariableDataPrototype` is referenced from a `NonqueuedReceiverComSpec` in the role `dataElement` and the attribute `aliveTimeout` of the `NonqueuedReceiverComSpec` exists and is set to a value  $> 0$  or
- the `VariableDataPrototype` is aggregated by a `SenderReceiverInterface` where attribute `invalidationPolicy.handleInvalid` exists and is set to the value `keep`.

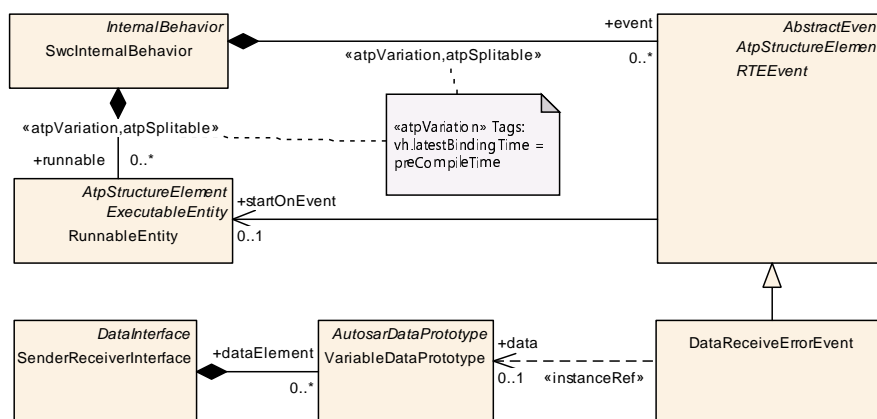
]()

**[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*)



**Figure 7.26: DataReceiveErrorEvent references a Runnable and a VariableDataPrototype**

**[TPS\_SWCT\_01341] DataReceiveErrorEvent is directly associated with the corresponding VariableDataPrototype** [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*)

This aspect is depicted in Figure 7.26.

## 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 7.27 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` **at the time when the contract phase generation is executed.**] ()

**[constr\_1775] Value of attribute `serverCallPoint.returnValueProvision`** [All `RunnableEntity.serverCallPoint` that refer to the same operation shall define the identical value of attribute `returnValueProvision` **at the time when the contract phase generation is executed.**] ()

Rationale for the existence of [constr\_1775]: different `RunnableEntities` could aggregate `ServerCallPoint` with a different configuration of attribute `returnValueProvision`.

However, in such a case it would not be possible to generate the corresponding RTE API because the API exists once per software-component and it is therefore indispensable to have all affected `ServerCallPoints` agree on the configuration of attribute `returnValueProvision`.

This relation is exemplarily (for the case of `dataSendPoint.returnValueProvision`) sketched in Figure 7.21.

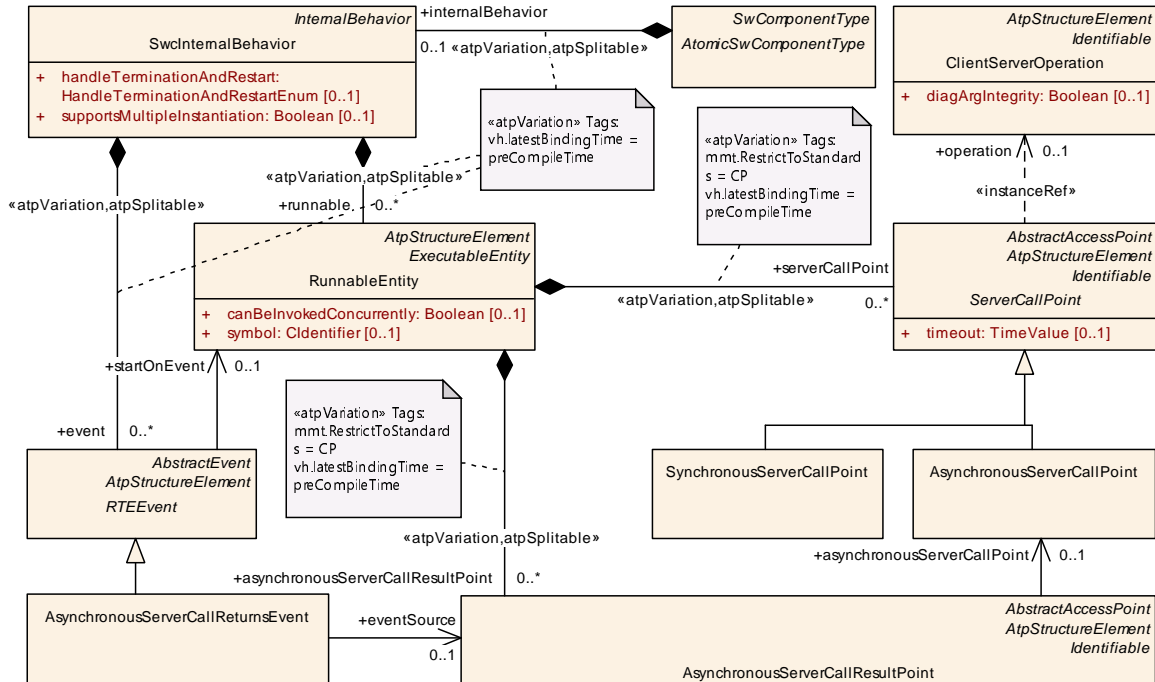
**[constr\_2023] Consistency of `timeout` values** [The `timeout` values of all `ServerCallPoints` referencing the same instance of `ClientServerOperation`



in a `RPortPrototype` shall be identical **at the time when the RTE is generated.**]

**[TPS\_SWCT\_01345] Synchronous operation invocation** [In case of a synchronous operation invocation the particular `RunnableEntity` merely needs a `SynchronousServerCallPoint`.] (*RS\_SWCT\_00200*)

More information can be found in Figure 7.27.



**Figure 7.27: Model of a server call point.**

**[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*)

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 `RunnableEntities`: 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 `PortPrototype` '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.

**[constr\_1776] Value of attribute `asynchronousServerCallResultPoint.returnValueProvision`** [All `RunnableEntity.asynchronousServerCallResultPoint` that refer to the same `AsynchronousServerCallPoint.operation` shall define the identical value of attribute `returnValueProvision` at the time when the contract phase generation is executed.]()

Rationale for the existence of [constr\_1776]: different `RunnableEntities` could aggregate `AsynchronousServerCallResultPoint` with a different configuration of attribute `returnValueProvision`.

However, in such a case it would not be possible to generate the corresponding RTE API because the API exists once per software-component and it is therefore indispensable to have all affected `AsynchronousServerCallResultPoints` agree on the configuration of attribute `returnValueProvision`.

This relation is exemplarily (for the case of `dataSendPoint.returnValueProvision`) sketched in Figure 7.21.

<b>Class</b>	<b>ServerCallPoint</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
<b>Note</b>	If a RunnableEntity owns a ServerCallPoint it is entitled to invoke a particular ClientServerOperation of a specific RPortPrototype of the corresponding AtomicSwComponentType			
<b>Base</b>	ARObject, AbstractAccessPoint, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
<b>Subclasses</b>	AsynchronousServerCallPoint, SynchronousServerCallPoint			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
operation	ClientServerOperation	0..1	iref	The operation that is called by this runnable. <b>InstanceRef implemented by:</b> ROperationInAtomicSw InstanceRef
timeout	TimeValue	0..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 7.37: ServerCallPoint**

**[constr\_1955] Existence of attribute `ServerCallPoint.operation`** [For each `ServerCallPoint`, attribute `operation` shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1956] Existence of attribute `ServerCallPoint.timeout`** [For each `ServerCallPoint`, attribute `timeout` shall exist **at the time when the RTE is generated.**]()

<b>Class</b>	<b>SynchronousServerCallPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
<b>Note</b>	This means that the RunnableEntity is supposed to perform a blocking wait for a response from the server.			
<b>Base</b>	ARObject, AbstractAccessPoint, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, ServerCallPoint			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
calledFrom WithinExclusive Area	ExclusiveAreaNesting Order	0..1	ref	This indicates that the call point is located at the deepest level inside one or more ExclusiveAreas that are nested in the given order.

**Table 7.38: SynchronousServerCallPoint**

**[constr\_1957] Existence of attribute `AsynchronousServerCallResultPoint.asynchronousServerCallPoint`** [For each `AsynchronousServerCallResultPoint`, the reference to `AsynchronousServerCallPoint` in the role `asynchronousServerCallPoint` shall exist **at the time when the contract phase generation is executed.**]()

<b>Class</b>	<b>AsynchronousServerCallPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			





<b>Class</b>	<b>AsynchronousServerCallPoint</b>			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">AbstractAccessPoint</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServerCallPoint</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 7.39: AsynchronousServerCallPoint**

<b>Class</b>	<b>AsynchronousServerCallResultPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">AbstractAccessPoint</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
asynchronous ServerCallPoint	<a href="#">AsynchronousServerCallPoint</a>	0..1	ref	The referenced Asynchronous Server Call Point defines the asynchronous server call from which the results are returned.

**Table 7.40: AsynchronousServerCallResultPoint**

**[constr\_2006] Number of [AsynchronousServerCallResultPoint](#) referencing to one [AsynchronousServerCallPoint](#)** [The [AsynchronousServerCallPoint](#) may be referenced by at most one [AsynchronousServerCallResultPoint](#) **at the time when the contract phase generation is executed.**

If the reference exists, this means that only the [RunnableEntity](#) with this [AsynchronousServerCallResultPoint](#) can fetch the result of the asynchronous server invocation of this particular [AsynchronousServerCallPoint](#).]()

Please note that if an [AsynchronousServerCallPoint](#) is **not** referenced by an [AsynchronousServerCallResultPoint](#) this means that there is no operation result to fetch or the caller **is not interested** in the result.

This information might be used by the RTE generator to optimize the data consistency mechanisms.

**[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.

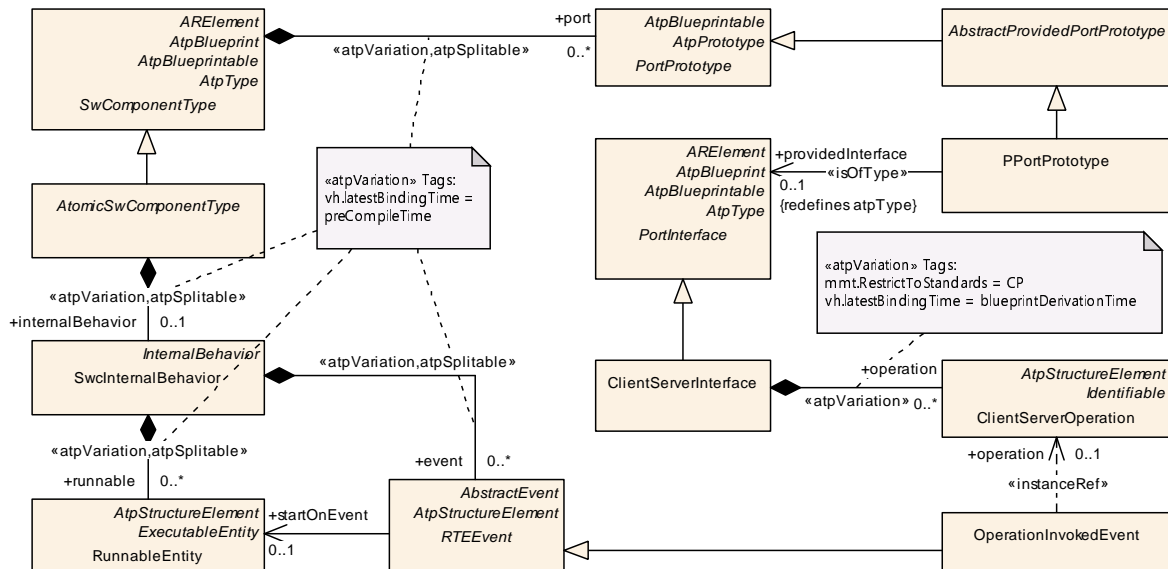
In this case the `AsynchronousServerCallReturnsEvent` shall not define a `startOnEvent` reference to a `RunnableEntity`.<sub>](RS\_SWCT\_00200)</sub>

**[constr\_2030] AsynchronousServerCallResultPoint combined with WaitPoint** shall belong to the same `RunnableEntity` [A `WaitPoint` referencing a `AsynchronousServerCallReturnsEvent` as well as a `AsynchronousServerCallResultPoint` referenced by said `AsynchronousServerCallReturnsEvent` shall be aggregated by the same `RunnableEntity` at the time when the contract phase generation is executed.]()

**[constr\_1521] Reference from AsynchronousServerCallReturnsEvent to AsynchronousServerCallResultPoint** [In the context of a `RunnableEntity`, a given `AsynchronousServerCallResultPoint` shall only be referenced by one `AsynchronousServerCallReturnsEvent` in the role `eventSource` at the time when the contract phase generation is executed.]()

### 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 7.28).



**Figure 7.28:** The `OperationInvokedEvent` references the operation that was called by a client.

### 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](#))

## 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`.

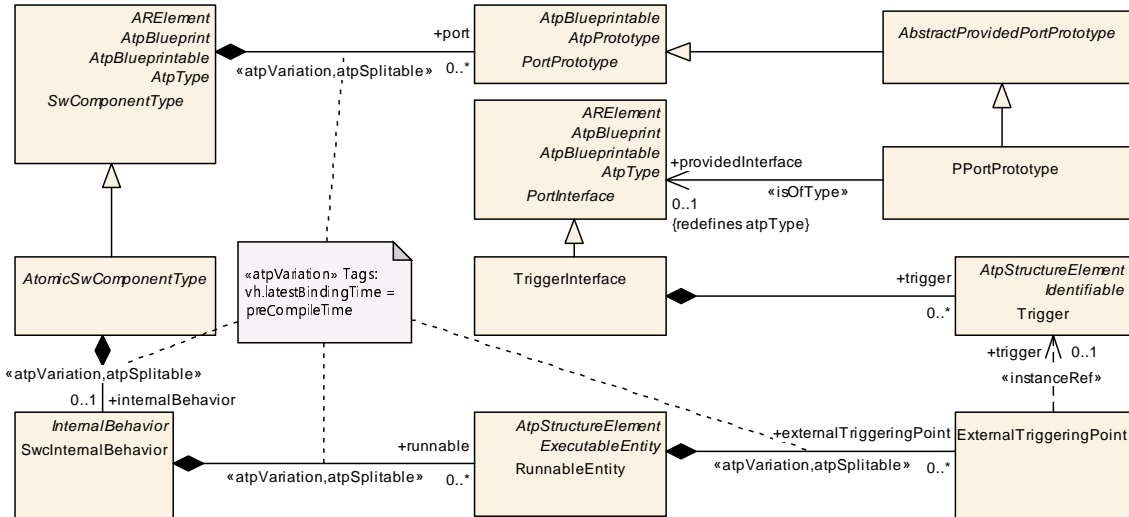
For this purpose the particular `RunnableEntity` needs an `ExternalTriggeringPoint` that references the particular instance of the trigger in a `PPortPrototype`.] ([RS\\_SWCT\\_00200](#))

**[constr\_1777] Value of attribute `externalTriggeringPoint.returnValueProvision`** [All `RunnableEntity.externalTriggeringPoint` that refer to the same `trigger` shall define the identical value of attribute `returnValueProvision` at the time when the contract phase generation is executed.] ()

Rationale for the existence of [\[constr\\_1777\]](#): different `RunnableEntities` could aggregate `ExternalTriggeringPoints` with a different configuration of attribute `returnValueProvision`.

However, in such a case it would not be possible to generate the corresponding RTE API because the API exists once per software-component and it is therefore indispensable to have all affected `ExternalTriggeringPoints` agree on the configuration of attribute `returnValueProvision`.

This relation is exemplarily (for the case of `dataSendPoint.returnValueProvision`) sketched in Figure 7.21.



**Figure 7.29: Model structure of a trigger source.**

<b>Class</b>	<b>ExternalTriggeringPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Trigger			
<b>Note</b>	If a RunnableEntity owns an ExternalTriggeringPoint it is entitled to raise an ExternalTriggerOccurred Event.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
ident	ExternalTriggeringPoint Ident	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).  <b>Stereotypes:</b> atpIdentityContributor <b>Tags:</b> xml.sequenceOffset=-100
trigger	Trigger	0..1	iref	The trigger taken for the ExternalTriggeringPoint.  <b>Tags:</b> xml.namePlural=TRIGGER-IREF xml.roleElement=false xml.roleWrapperElement=true xml.typeElement=true xml.typeWrapperElement=false <b>InstanceRef implemented by:</b> PTriggerInAtomicSwcType InstanceRef

**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.

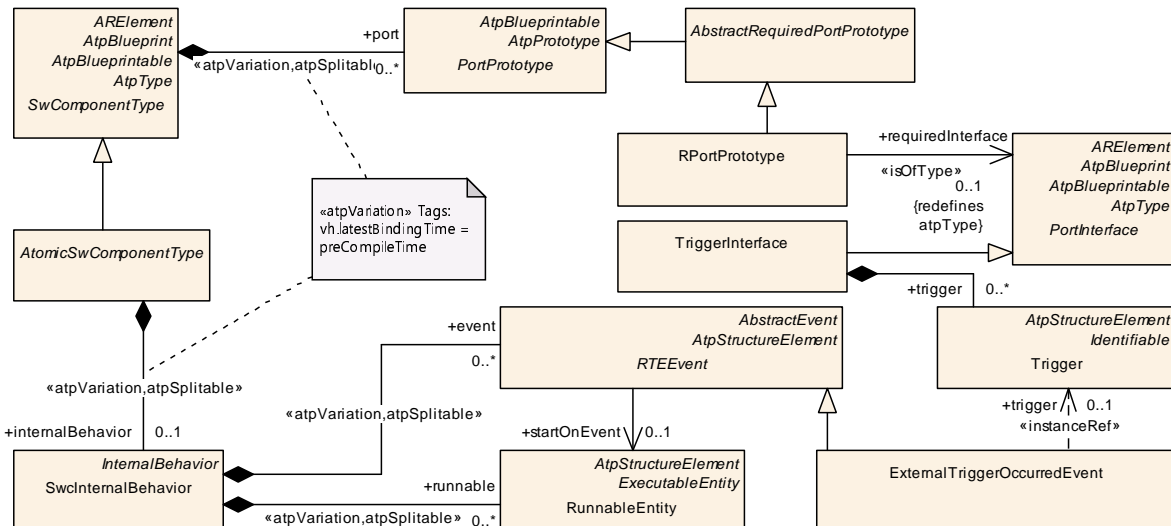


Figure 7.30: Model structure of a trigger sink

**[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*)

### 7.5.4 RunnableEntities and Parameter Access

There are several ways a Calibration Parameter is provided within a software component.

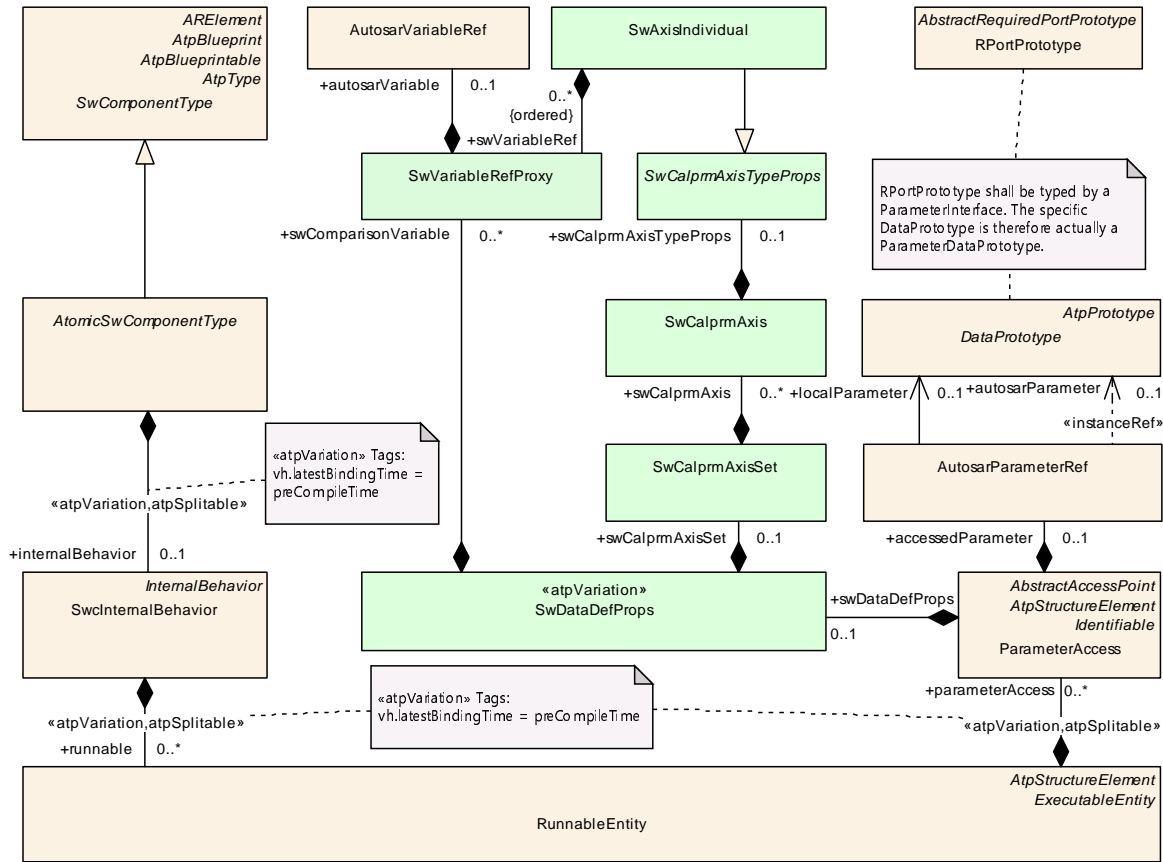
**[TPS\_SWCT\_01350] Calibration Parameters shared among several SwComponentTypes** [As mentioned above, if Calibration Parameters are shared among several `SwComponentTypes` a dedicated `PortInterface` in a `PortPrototype` will be used.] (*RS\_SWCT\_00200*)

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 an adjacent `SwComponentPrototype` via dedicated `PortPrototypes`.





**Figure 7.31: Runnable Access to a Calibration Port**

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`.

Class	ParameterAccess			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
Note	The presence of a <code>ParameterAccess</code> implies that a <code>RunnableEntity</code> needs access to a <code>ParameterDataPrototype</code> .			
Base	<code>ARObject</code> , <code>AbstractAccessPoint</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code>			
Attribute	Type	Mult.	Kind	Note
accessedParameter	<code>AutosarParameterRef</code>	0..1	aggr	Reference to the accessed calibration parameter.
swDataDefProps	<code>SwDataDefProps</code>	0..1	aggr	This allows denote instance and access specific properties, mainly input values and common axis.

**Table 7.42: ParameterAccess**

**[constr\_1958] Existence of attribute `ParameterAccess.accessedParameter`**  
[For each `ParameterAccess`, attribute `accessedParameter` shall exist **at the time when the contract phase generation is executed.** ]()

**[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`.
- by defining the `ParameterAccess` association from a `RunnableEntity` to the `ParameterDataPrototype` in the roles `sharedParameter` or `perInstanceParameter`.

] (*RS\_SWCT\_00200*)

Please find more information about the topic of [TPS\_SWCT\_01351] in Figure 7.31 as well as in Figure 2.3 in the lower association from `RunnableEntity` to `ParameterDataPrototype`

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.

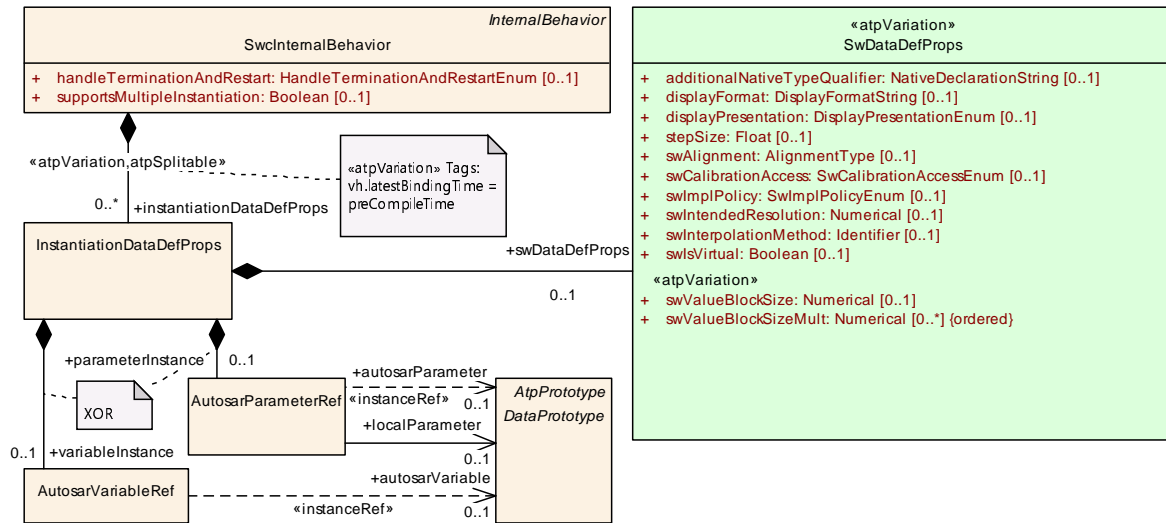
### 7.5.4.1 InstantiationDataDefProps

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 re-use 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 7.32



**Figure 7.32: applying instantiation specific data definition properties**

<b>Class</b>	<b>InstantiationDataDefProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::InstantiationDataDefProps			
<b>Note</b>	<p>This is a general class allowing to apply additional SwDataDefProps to particular instantiations of a Data Prototype.</p> <p>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 Application CompositeDataType).</p> <p>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).</p> <p>This class overcomes such a restriction if applied properly.</p>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
parameter Instance	AutosarParameterRef	0..1	aggr	This is the particular ParameterDataPrototypes on which the swDataDefProps shall be applied.
swDataDef Props	SwDataDefProps	0..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 7.43: InstantiationDataDefProps**

**[constr\_1959] Existence of attribute InstantiationDataDefProps.swDataDefProps** [For each InstantiationDataDefProps, attribute swDataDefProps shall exist at the time when the contract phase generation is executed.]()

### 7.5.5 RunnableEntities and Mode Communication

For the communication of modes between `RunnableEntity`s 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 `RunnableEntity`s 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`.] ([RS\\_SWCT\\_00200](#))

This concept and the meta-classes needed for the mapping are further explained in chapter [4.2.5](#).

**[TPS\_SWCT\_01353] RunnableEntities react on a mode request via a corresponding RTEEvent** [In the second case, one `RunnableEntity` “sends” a mode request and one or more other `RunnableEntity`s 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.] ([RS\\_SWCT\\_00200](#), [RS\\_SWCT\\_03202](#))

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](#).

## 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.] ([RS\\_SWCT\\_03040](#))

Please note that meta-class `PortAPIOption` is depicted in Figure [7.33](#).

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 Runnable Entity to the PortPrototype).			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
enableTakeAddress	Boolean	0..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 a RunnableEntity accessing a Port Prototype that is referenced by this PortAPIOption shall specifically handle transformer errors or not.
indirectAPI	Boolean	0..1	attr	If set to true this attribute specifies an "indirect API" to be generated for the associated port which means that the software-component 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	0..1	ref	The option is valid for generated functions related to communication over this port
portArgValue (ordered)	PortDefinedArgumentValue	*	aggr	An argument value defined by this port.
supportedFeature	SwcSupportedFeature	*	aggr	This collection specifies which features are supported by the RunnableEntitys which access a PortPrototype that it referenced by this PortAPIOption.
transformerStatusForwarding	DataTransformationStatusForwardingEnum	0..1	attr	This specifies whether a RunnableEntity accessing a Port Prototype that is referenced by this PortAPIOption shall be able to forward a status to the transformer chain.

Table 7.44: PortAPIOption

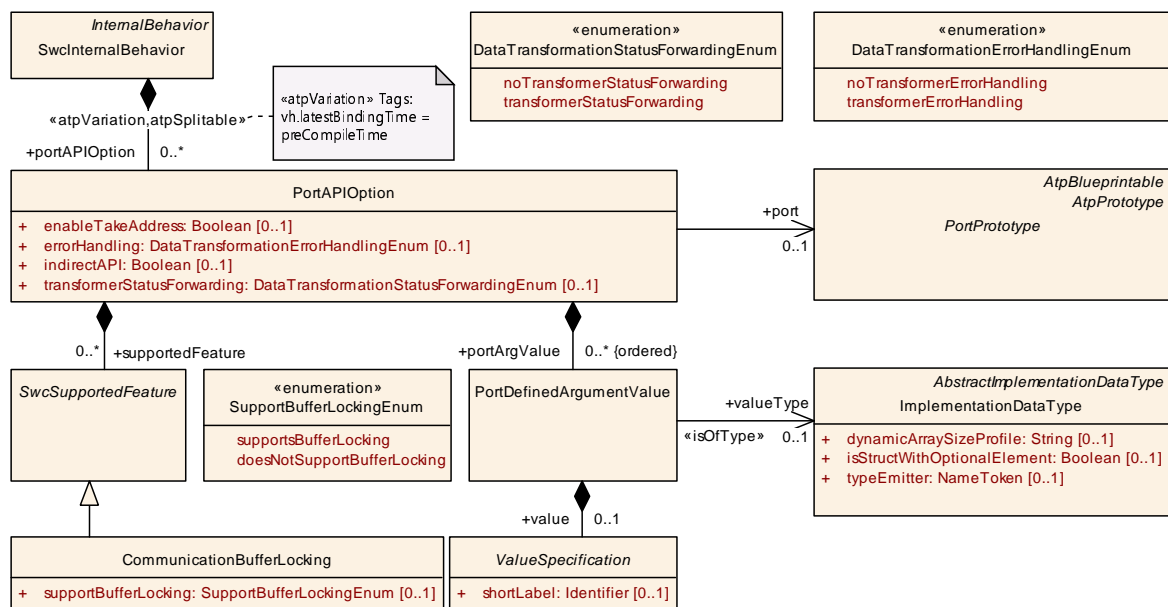


Figure 7.33: Port API Options.

[constr\_1960] Existence of attribute `PortAPIOption.port` [For each `PortAPIOption`, attribute `port` shall exist at the time when the contract phase generation is executed.]()

[TPS\_SWCT\_01626] **Error notification of data transformer errors** [If the attribute `PortAPIOption.errorHandling` is set to `transformerErrorHandling` then all `RunnableEntity`s accessing the `PortPrototype` referenced by `port` shall handle the extended transformer error notification.] ([RS\\_SWCT\\_03222](#), [RS\\_SWCT\\_03221](#))

[TPS\_SWCT\_03500] **Status forwarding to data transformer** [If the attribute `PortAPIOption.transformerStatusForwarding` is set to `transformerStatusForwarding` then all `RunnableEntity`s accessing the `PortPrototype` referenced by `port` shall provide the to be forwarded transformer status.] ([RS\\_SWCT\\_03040](#), [RS\\_SWCT\\_03221](#))

Enumeration	DataTransformationErrorHandlingEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions
Note	This enumeration defines different ways how a <code>RunnableEntity</code> shall handle transformer errors.
Literal	Description
noTransformerErrorHandling	A runnable does not handle transformer errors. Tags:atp.EnumerationLiteralIndex=0
transformerErrorHandling	The runnable implements the handling of transformer errors. Tags:atp.EnumerationLiteralIndex=1

**Table 7.45: DataTransformationErrorHandlingEnum**

[TPS\_SWCT\_03501] **Applicability of status forwarding to data transformer** [The attribute `PortAPIOption.transformerStatusForwarding` shall only be set to `transformerStatusForwarding` if the transported data is originating from a transformed source and is not altered.] ([RS\\_SWCT\\_03040](#), [RS\\_SWCT\\_03221](#))

The restriction of [TPS\_SWCT\_03501] is especially necessary in case of E2E protected data routing where the source of the data is received with a *transformer error* using `PortAPIOption.errorHandling = transformerErrorHandling`.

Enumeration	DataTransformationStatusForwardingEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions
Note	This enumeration defines different ways how a <code>RunnableEntity</code> shall be able to forward status into the transformer chain.
Literal	Description
noTransformerStatusForwarding	The runnable is not able to forward a transformer status. Tags:atp.EnumerationLiteralIndex=0
transformerStatusForwarding	The runnable is able to forward a transformer status. Tags:atp.EnumerationLiteralIndex=1

**Table 7.46: DataTransformationStatusForwardingEnum**

### 7.6.1 Enable to Take Address

[TPS\_SWCT\_01355] **enableTakeAddress = true** [If the attribute `enableTakeAddress = true` the generated API functions related to this `PortPrototype` shall be

implemented by means of true/native C functions (as opposed to function-like preprocessor macros) so that it is possible to access the API functions via their address (by means of function-pointers).|(RS\_SWCT\_03040)

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.supportMultipleInstantiation` is set to `false` **at the time when the contract phase generation is executed.**]|()

## 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 iterating over ports within the SW-Component.](RS\_SWCT\_03040)

## 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.](RS\_SWCT\_03040)

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 of the server component details.](RS\_SWCT\_03040)

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 7.33 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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

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` **at the time when the RTE is generated.**]()

To be clear, this means that `PortDefinedArgumentValues` may not be used together with `RPortPrototypes`.

<b>Class</b>	<b>PortDefinedArgumentValue</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions			
<b>Note</b>	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> .			
<b>Base</b>	<code>ARObject</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
value	<code>ValueSpecification</code>	0..1	aggr	Specifies the actual value.
valueType	<code>ImplementationDataType</code>	0..1	tref	The implementation type of this argument value. It should not be composite type or a pointer. <b>Stereotypes:</b> <code>isOfType</code>

**Table 7.47: PortDefinedArgumentValue**

**[constr\_1961] Existence of attribute `PortDefinedArgumentValue.value`** [For each `PortDefinedArgumentValue`, attribute `value` shall exist **at the time when the RTE is generated.**]()

**[constr\_1962] Existence of attribute `PortDefinedArgumentValue.valueType`** [For each `PortDefinedArgumentValue`, attribute `valueType` shall exist **at the time when the RTE is generated.**]()

## 7.6.4 Supported Features

Historically, the `PortAPIOption` has undergone a number of extensions that usually ended up in additional primitive or composite attributes.



As further requests for extensions keep coming in, focus was put on limiting the complexity of the overall modeling of `PortAPIOption`. In response to this, a new extension approach has been defined to keep the surroundings of `PortAPIOption` manageable.

In particular, `PortAPIOption` aggregates the abstract meta-class `SwcSupportedFeature` in the role `supportedFeature` (see Figure 7.33).

The actual aggregation of `supportedFeature` will consist of concrete sub-classes of `SwcSupportedFeature`.

It will be possible to add further sub-classes of `SwcSupportedFeature` to add further functionality without increasing the modeling complexity of `PortAPIOption`, at the expense of having to formulate additional constraints.

<b>Class</b>	<code>SwcSupportedFeature</code> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions			
<b>Note</b>	This meta-class represents a abstract base class for features that can be supported by a <code>RunnableEntity</code> .			
<b>Base</b>	<code>ARObject</code>			
<b>Subclasses</b>	<code>CommunicationBufferLocking</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 7.48: SwcSupportedFeature**

### 7.6.4.1 Buffer Locking

**[TPS\_SWCT\_01687] Support of locked communication buffers** [If a `CommunicationBufferLocking` where attribute `supportBufferLocking` is set to value `supportsBufferLocking` is aggregated in the role `PortAPIOption.supportedFeature` then all `RunnableEntities` accessing the enclosing `PortPrototype` shall be able to support the return value `RTE_E_COM_BUSY`.] (*RS\_SWCT\_03040*)

**[constr\_1432] Multiplicity of `CommunicationBufferLocking`** [In a concrete aggregated set of `PortAPIOption.supportedFeature`, `CommunicationBufferLocking` shall exist **at most once at the time when the RTE is generated**.] ()

<b>Class</b>	<code>CommunicationBufferLocking</code>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions			
<b>Note</b>	The aggregation of this meta-class specifies that a <code>RunnableEntity</code> supports locked communication buffers supplied by the RTE. It is able to cope with the error <code>RTE_E_COM_BUSY</code> .			
<b>Base</b>	<code>ARObject</code> , <code>SwcSupportedFeature</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
<code>supportBufferLocking</code>	<code>SupportBufferLockingEnum</code>	0..1	attr	This attribute is used to indicate the intended buffer locking behavior.

**Table 7.49: CommunicationBufferLocking**

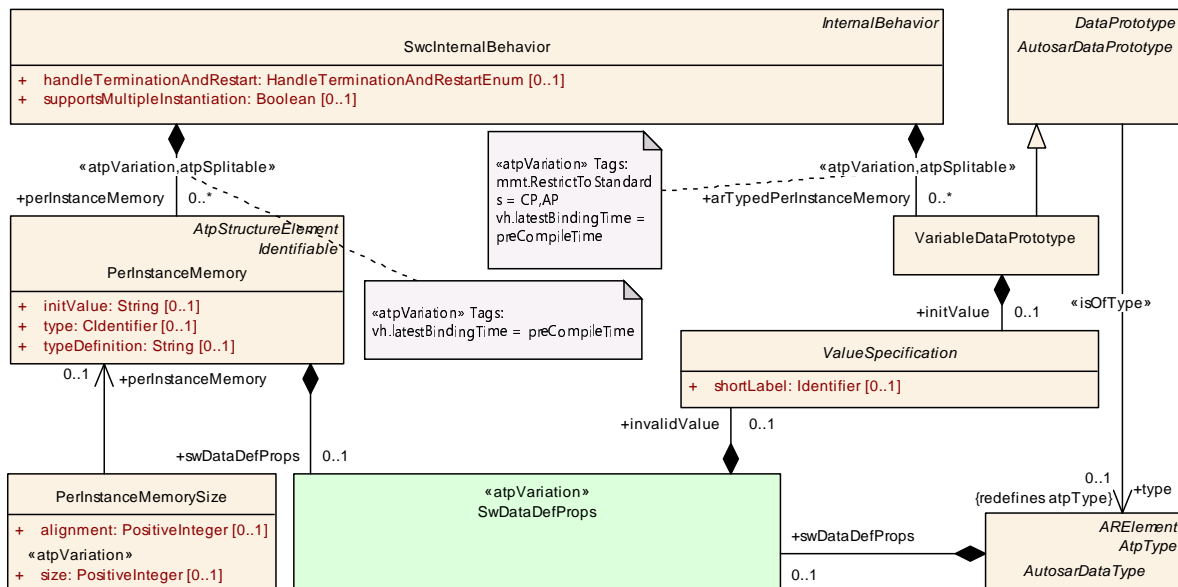
**[constr\_1963] Existence of attribute `CommunicationBufferLocking.support-BufferLocking`** [For each `CommunicationBufferLocking`, attribute `support-BufferLocking` shall exist **at the time when the RTE is generated.**]([\(\)](#))

Enumeration	SupportBufferLockingEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions
Note	This enumeration represents the ability to define the buffer locking behavior.
Literal	Description
doesNotSupportBufferLocking	Buffer locking is not supported. <b>Tags:</b> atp.EnumerationLiteralIndex=0
supportsBufferLocking	Buffer locking is supported. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 7.50: SupportBufferLockingEnum**

## 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 mechanism with which each instance of an `AtomicSwComponentType` can access its own instance-specific memory.]([RS\\_SWCT\\_03040](#))



**Figure 7.34: PerInstanceMemory**

**[TPS\_SWCT\_01360] Arbitrary number of per-instance memory blocks** [An `AtomicSwComponentType` can define an arbitrary number of per-instance memory blocks.]([RS\\_SWCT\\_03040](#))

[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.](RS\_SWCT\_03040)

[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.](RS\_SWCT\_03040)

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.

This rule shall be imposed **at the time when the contract phase generation is executed**.]()

[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].

<b>Class</b>	<b>PerInstanceMemory</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PerInstanceMemory			
<b>Note</b>	Defines a 'C' typed memory-block that needs to be available for each instance of the SW-component. This is typically only useful if <code>supportsMultipleInstantiation</code> is set to "true" or if the software-component defines NVRAM access via permanent blocks.			
<b>Base</b>	<i>ARObject</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
<code>initValue</code>	<code>String</code>	0..1	attr	Specifies initial value(s) of the <code>PerInstanceMemory</code>
<code>swDataDef Props</code>	<code>SwDataDefProps</code>	0..1	aggr	This represents the ability to to allocate RAM at specific memory sections, for example, to support the RAM Block recovery strategy by mapping to uninitialized RAM.
<code>type</code>	<code>CIdentifier</code>	0..1	attr	The name of the "C"-type
<code>typeDefinition</code>	<code>String</code>	0..1	attr	A definition of the type with the syntax of a 'C' typedef.

**Table 7.51: PerInstanceMemory**

**[constr\_1964] Existence of attribute `PerInstanceMemory.type`** [For each `PerInstanceMemory`, attribute `type` shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1965] Existence of attribute `PerInstanceMemory.typeDefinition`** [For each `PerInstanceMemory`, attribute `typeDefinition` shall exist **at the time when the contract phase generation is executed.**]()

## 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`](*RS\_SWCT\_03040*)

**[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 an MCD system via a `SwDataDefProps` for the purpose of measurement or defining an input value of an axis.] (RS\_SWCT\_03040)

For more information about the relevance for measurement please refer to chapter 5.4.3. The aspect of defining an input value of an axis is explained in 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

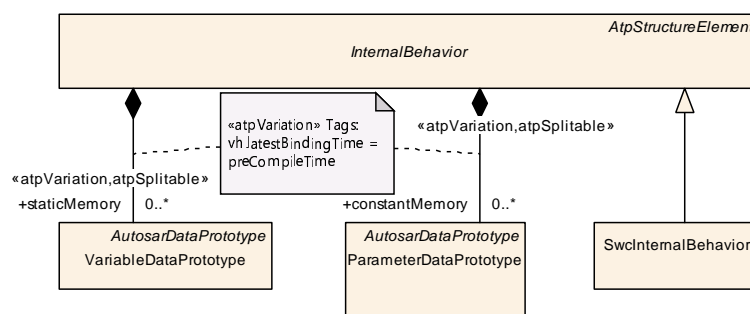
**[TPS\_SWCT\_01368] Describe static and constant memory** [Static memory (formalized by means of `InternalBehavior.staticMemory`) and constant memory (formalized by means of `InternalBehavior.constantMemory`) can be used whenever `AutosarDataTypes` should be used in the implementation of an `AtomicSwComponentType` but no involvement of the RTE (for memory allocation and management) is required.] (RS\_SWCT\_03040)

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 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.] (RS\_SWCT\_03040)

Please note that the topic “measurement and calibration” is discussed in chapter 2.2.



**Figure 7.35: 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. *(RS\_SWCT\_03040)*

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` **at the time when the RTE is generated**.] *()*

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 or 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.] *(RS\_SWCT\_03040)*

**[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.] *(RS\_SWCT\_03040)*

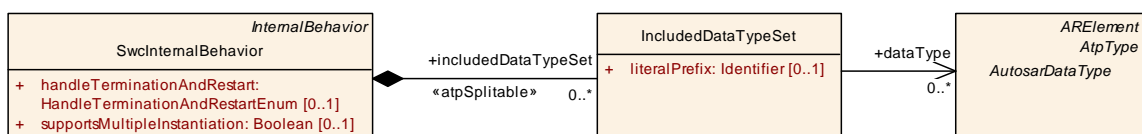


Figure 7.36: Included AUTOSAR Data Types

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.

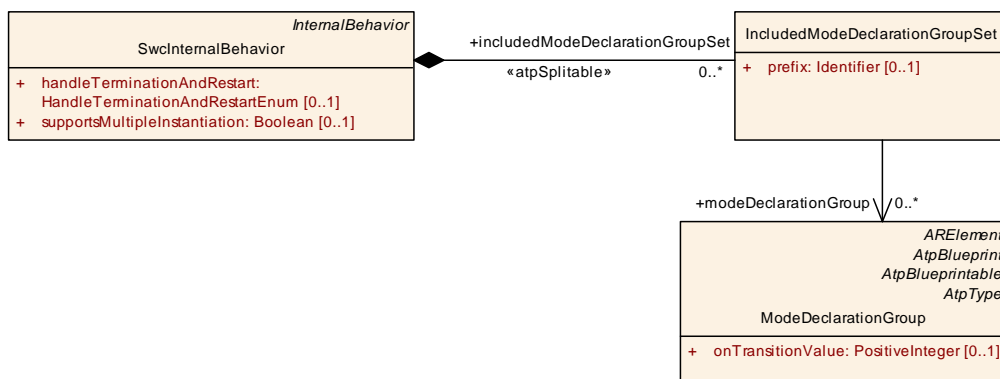
<b>Class</b>	<b>IncludedDataTypeSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::IncludedDataTypes			
<b>Note</b>	<p>An includedDataTypeSet declares that a set of AutosarDataType is used by a basic software module or a software component for its implementation and the AutosarDataType becomes part of the contract.</p> <p>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.</p> <p>The optional literalPrefix is used to add a common prefix on enumeration literals, lowerLimit and upperLimit constants created by the RTE.</p>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataType	AutosarDataType	*	ref	AutosarDataType belonging to the includedDataTypeSet
literalPrefix	Identifier	0..1	attr	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 7.52: IncludedDataTypeSet**

[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.] ([RS\\_SWCT\\_03040](#))

## 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**.] ([RS\\_SWCT\\_03040](#), [RS\\_SWCT\\_03110](#))



**Figure 7.37: Included ModeDeclarationGroups**

<b>Class</b>	<b>IncludedModeDeclarationGroupSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ModeDeclarationGroup			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
mode Declaration Group	<a href="#">ModeDeclarationGroup</a>	*	ref	This represents the referenced ModeDeclarationGroup.
prefix	Identifier	0..1	attr	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 7.53: 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.]  
 ([RS\\_SWCT\\_03040](#), [RS\\_SWCT\\_03110](#))

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](#) 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](#))

Please note that some [ServiceNeeds](#) are on display in Figures [7.38](#), [13.9](#), [13.6](#), and [13.11](#).

**[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 [33].

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 [6].

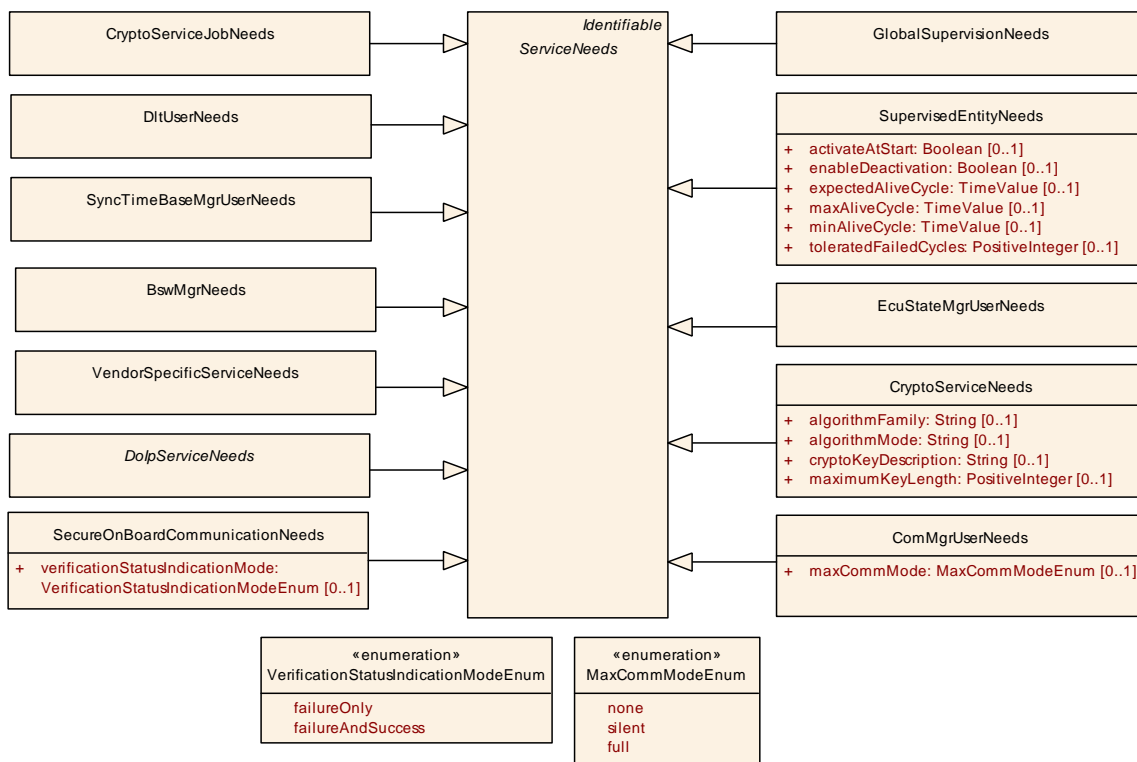


Figure 7.38: Meta-class *ServiceNeeds* and some of its specializations

The semantics of sub-classes of *ServiceNeeds* is explained in the respective sub-chapters of chapter 13.

<b>Class</b>	<i>ServiceNeeds</i> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	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.
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>





<b>Class</b>	<b>ServiceNeeds</b> (abstract)			
<b>Subclasses</b>	BswMgrNeeds, ComMgrUserNeeds, CryptoKeyManagementNeeds, CryptoServiceJobNeeds, CryptoServiceNeeds, <i>DiagnosticCapabilityElement</i> , DltUserNeeds, <i>DolpServiceNeeds</i> , EcuStateMgrUserNeeds, ErrorTracerNeeds, FunctionInhibitionAvailabilityNeeds, FunctionInhibitionNeeds, GlobalSupervisionNeeds, HardwareTestNeeds, IdsMgrCustomTimestampNeeds, IdsMgrNeeds, IndicatorStatusNeeds, J1939DcmDm19Support, J1939RmIncomingRequestServiceNeeds, J1939RmOutgoingRequestServiceNeeds, NvBlockNeeds, SecureOnBoardCommunicationNeeds, SupervisedEntityCheckpointNeeds, SupervisedEntityNeeds, SyncTimeBaseMgrUserNeeds, V2xFacUserNeeds, V2xMUserNeeds, VendorSpecificServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 7.54: ServiceNeeds**

Please note that the vast majority of the subclasses of meta-class `ServiceNeeds` are associated with standardized behavior of AUTOSAR services. However, there are cases where a user-specific behavior is required and for this purpose a specific flavor of `ServiceNeeds` is available.

**[TPS\_SWCT\_01693] Usage of `VendorSpecificServiceNeeds`** [It is possible to define `VendorSpecificServiceNeeds` for the purpose of implementing a vendor-specific, i.e. non-standardized, service. `VendorSpecificServiceNeeds` does not provide any attributes and its meaning shall be described by means of the `category` attribute.] ([RS\\_SWCT\\_02060](#))

<b>Class</b>	<b>VendorSpecificServiceNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This represents the ability to define vendor-specific service needs.			
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 7.55: VendorSpecificServiceNeeds**

### 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 [AtomicSwComponentTypes](#) 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](#).

<b>Class</b>	<b>RoleBasedPortAssignment</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
<b>Note</b>	This class specifies an assignment of a role to a particular service port (RPortPrototype or PPort Prototype) 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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
portPrototype	<a href="#">PortPrototype</a>	0..1	ref	Service PortPrototype used in the assigned role. This PortPrototype shall either belong to the same AtomicSw ComponentType as the SwcInternalBehavior which owns the ServiceDependency or to the same NvBlockSw ComponentType as the NvBlockDescriptor.
role	Identifier	0..1	attr	This is the role of the assigned Port in the given context. The value shall be a shortName of the Blueprint of a Port Interface as standardized in the Software Specification of the related AUTOSAR Service.

**Table 7.56: RoleBasedPortAssignment**

Both are aggregating an attribute [role](#) which allows to define the role of the [Port-Prototypes](#) 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](#) or via [assignedData](#) shall be typed by a [PortInterface](#) that has [isService](#) set to true **at the time when the RTE is generated.**

This rule does **not** apply to [PortPrototypes](#) referenced by a [RoleBasedPortAssignment](#) where the attribute [role](#) is set to any of the following values:

- NvMService
- NvMNotifyJobFinished
- NvMNotifyInitBlock
- NvMAdmin
- NvMMirror
- NvDataPort

Furthermore, the rule does **not** apply to the case described in [TPS\_SWCT\_01579], [TPS\_SWCT\_01831], [TPS\_SWCT\_01580], and [TPS\_SWCT\_01572].] ()

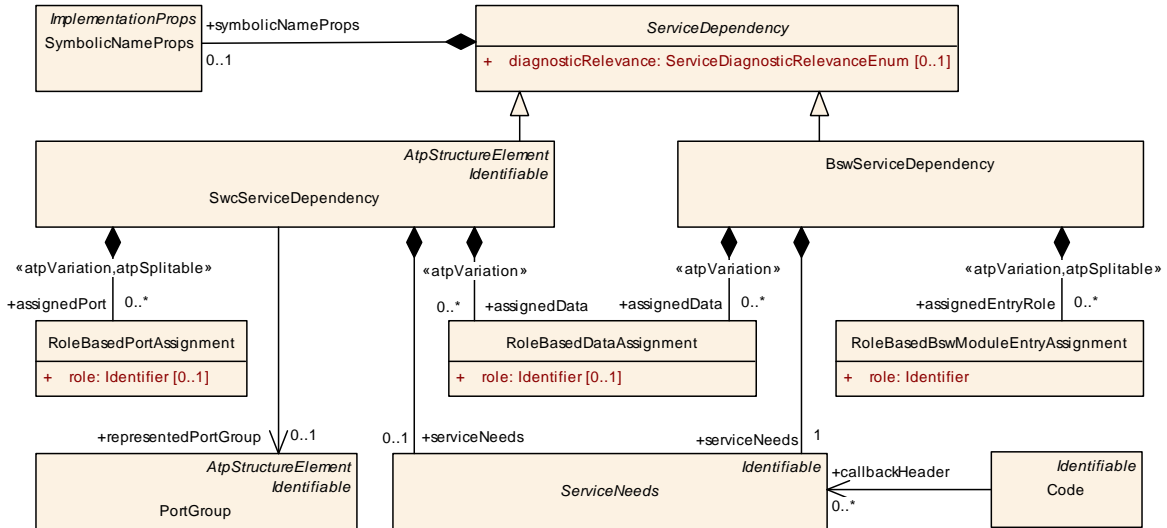


Figure 7.39: ServiceDependency is the abstract base class of SwcServiceDependency

The actual mapping between the ServiceNeeds element and its various relationships is provided by the meta-class SwcServiceDependency as shown in figure 7.40.

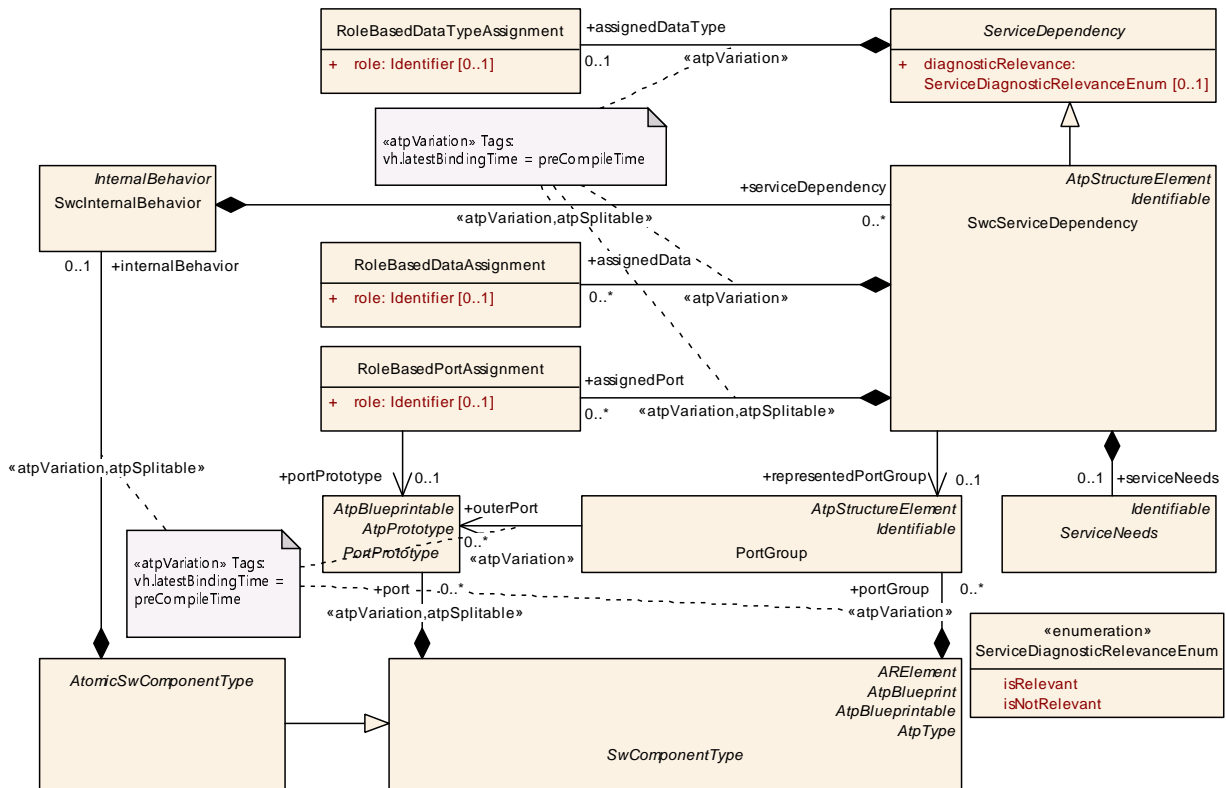


Figure 7.40: SwcServiceDependency in the SwcInternalBehavior

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`.

This means a `PortGroup` represents a system feature, whereas the `RoleBasedPortAssignment` is a local feature for the purpose of communication with the AUTOSAR Service.

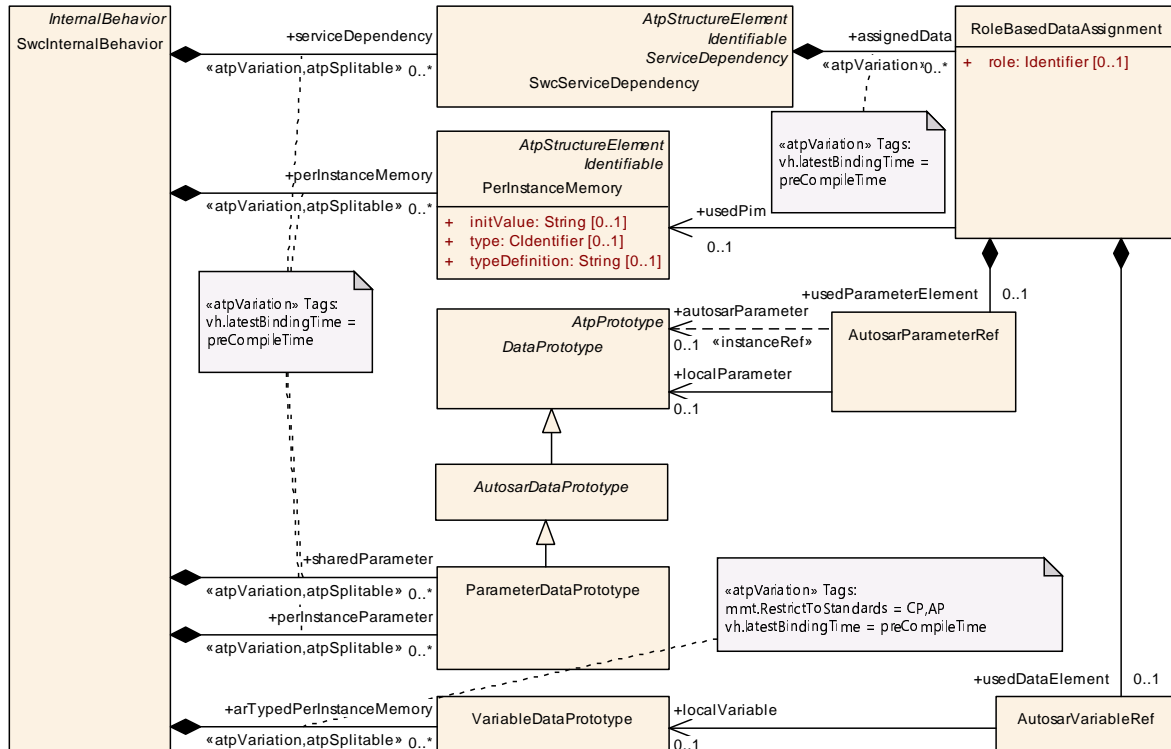


Figure 7.41: Details of `RoleBasedDataAssignment` for local data

<b>Class</b>	<b>RoleBasedDataAssignment</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	<p>This class specifies an assignment of a role to a particular data object in either</p> <ul style="list-style-type: none"> <li>the <code>SwcInternalBehavior</code> of a software component (or in the <code>BswInternalBehavior</code> of a BSW module or BSW cluster) in the context of an AUTOSAR Service or</li> <li>an <code>NvBlockDescriptor</code> to sort out the assignment of event-based writing strategies to data elements in a <code>PortPrototype</code>.</li> </ul> <p>With this assignment, the role of the data can be mapped to a <code>DataPrototype</code> that is used in the context of the definition of a specific <code>ServiceNeeds</code> or <code>NvBlockDescriptor</code>, so that a tool is able to create the correct access or writing strategy.</p>			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
role	Identifier	0..1	attr	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.



Class	RoleBasedDataAssignment			
				<p>△</p> <p>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.</p> <p>The following values are standardized:</p> <ul style="list-style-type: none"> <li>• <b>ramBlock</b> indicates data to be used as a mirror for an NVRAM Block.</li> <li>• <b>defaultValue</b> indicates constant data to be used as default in the context of this ServiceNeeds, e.g. for an NVRAM Block.</li> <li>• <b>signalBasedDiagnostics</b> indicates the Role BasedDataAssignment shall be used for signal based diagnostics.</li> </ul>
usedData Element	<a href="#">AutosarVariableRef</a>	0..1	aggr	<p>The VariableDataPrototype used in this role, e.g.</p> <ul style="list-style-type: none"> <li>• Permanent RAM Block of an NVRAM Block which shall belong to the same SwcInternalBehavior or BswInternalBehavior.</li> <li>• In the role signalBasedDiagnostics it has to refer to a VariableDataPrototype in a SenderReceiverInterface or a NvDataInterface.</li> </ul>
usedParameter Element	<a href="#">AutosarParameterRef</a>	0..1	aggr	<p>The ParameterDataPrototype used in this role, e.g.</p> <ul style="list-style-type: none"> <li>• ROM Block of an NVRAM Block. It shall belong to the same SwcInternalBehavior or BswInternalbehavior.</li> <li>• In the role signalBasedDiagnostics it has to refer to a ParameterDataPrototype in a ParameterInterface.</li> </ul>
usedPim	<a href="#">PerInstanceMemory</a>	0..1	ref	<p>The (untyped) PerInstanceMemory used in this role (e.g. as a Permanent RAM Block for an NVRAM Block).</p>

**Table 7.57: RoleBasedDataAssignment**

[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 BLUEPRINT<sup>6</sup> and the top-most [ARPackage.shortName](#) is set to AUTOSAR, see also [28]).] ([RS\\_SWCT\\_02060](#))

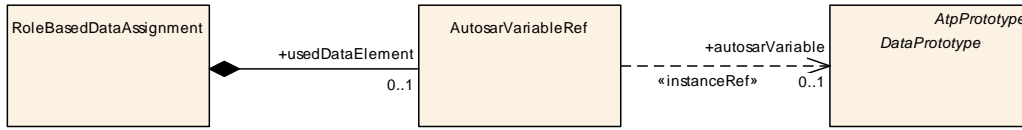
[TPS\_SWCT\_01660] Values of [SwcServiceDependency.category](#) reserved by the standard [The following values of [SwcServiceDependency.category](#) are reserved by the AUTOSAR standard:

**SERVICE** : this applies for all the cases where [SwcServiceDependency](#) is intended to be used for the design of [ServiceSwComponentTypes](#).

**NV\_BLOCK\_COMPONENT** : this applies if the [SwcServiceDependency](#) is intended to be used for the design of an [NvBlockSwComponentType](#).

] ([RS\\_SWCT\\_02060](#))

<sup>6</sup>see [TPS\_STDT\_00033]



**Figure 7.42: Details of RoleBasedDataAssignment for accessing DataPrototypes in PortPrototypes**

<b>Class</b>	<b>SwcServiceDependency</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
<b>Note</b>	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.			
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, ServiceDependency			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
assignedData	RoleBasedData Assignment	*	aggr	Defines the role of an associated data object of the same component. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
assignedPort	RoleBasedPort Assignment	*	aggr	Defines the role of an associated port of the same component. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=assignedPort, assignedPort.variation Point.shortLabel vh.latestBindingTime=preCompileTime
representedPort Group	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 Port Groups, 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	0..1	aggr	The associated ServiceNeeds.

**Table 7.58: SwcServiceDependency**

<b>Class</b>	<b>ServiceDependency</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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 (Service Needs) 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.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	BswServiceDependency, SwcServiceDependency			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
assignedData Type	RoleBasedDataType Assignment	0..1	aggr	This is the role of the assignment data type in the given context. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime





Class	ServiceDependency (abstract)			
diagnosticRelevance	ServiceDiagnosticRelevanceEnum	0..1	attr	If this attribute indicates a relevance for diagnostics then the integrator has a much easier time identifying the candidates for the configuration of the diagnostic stack.  Example: identification of mode conditions (e.g. communication between application and BswM) relevant for the Dcm.
symbolicNameProps	SymbolicNameProps	0..1	aggr	This attribute can be taken to contribute to the creation of symbolic name values.

**Table 7.59: ServiceDependency**

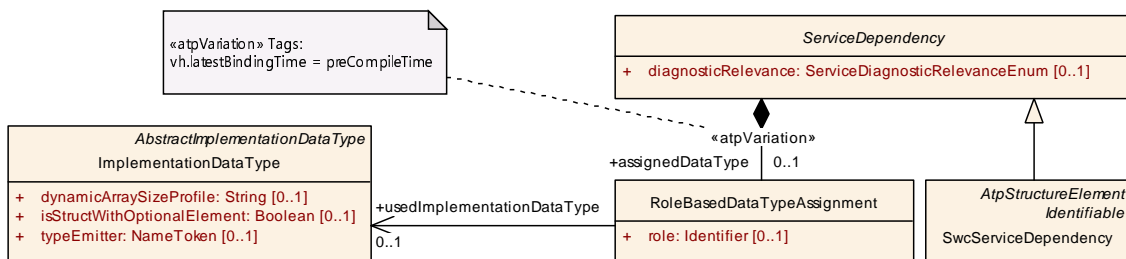
Enumeration	ServiceDiagnosticRelevanceEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	This enumeration provides values to describe the diagnostic relevance of a SwcServiceDependency (specifically if the aggregated ServiceNeeds itself does not indicate a relevance for diagnostics).
Literal	Description
isNotRelevant	This value indicates that a relevance for diagnostics does not exist. <b>Tags:</b> atp.EnumerationLiteralIndex=0
isRelevant	This value indicates a relevance for diagnostics. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 7.60: ServiceDiagnosticRelevanceEnum**

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	Type	Mult.	Kind	Note
-	-	-	-	-

**Table 7.61: SymbolicNameProps**

[TPS\_SWCT\_01661] Default value of **SwcServiceDependency.category** [If the attribute **SwcServiceDependency.category** does not exist then the value **SERVICE** shall be assumed for **SwcServiceDependency.category**.] (*RS\_SWCT\_02060*)



**Figure 7.43: Details of RoleBasedDataTypeAssignment for local data**



<b>Class</b>	<b>RoleBasedDataTypeAssignment</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
role	Identifier	0..1	attr	This is the role of the associated data type in the given context.
used Implementation DataType	<a href="#">ImplementationDataType</a>	0..1	ref	This represents the associated ImplementationDataType.

**Table 7.62: RoleBasedDataTypeAssignment**

**[constr\_10020] Existence of attribute [RoleBasedDataTypeAssignment.usedImplementationDataType](#)** [For each [RoleBasedDataTypeAssignment](#), attribute [usedImplementationDataType](#) shall exist **at the time when the RTE is generated.**]()

Please note that there are cases where the granularity of existing [ServiceInterfaces](#) does not match the granularity of existing [SwcServiceDependency.serviceNeeds](#).

In other words, there are [Service Interfaces](#) that cover the semantics of different kinds of [ServiceNeeds](#). One example is the [ClientServerInterface](#) `DataServices_{Data}` which basically supports the access to diagnostic values **as well as** I/O control of the same value.

Figure 7.44 provides a graphical sketch of how the modeling for this case is foreseen.

**[TPS\_SWCT\_01689] Relation between [SwcServiceDependencies](#) and [PortPrototypes](#)** [It is positively possible to create a model where two or more [SwcServiceDependencies](#), by way of the [RoleBasedPortAssignment](#) or [RoleBasedDataAssignment](#), refer to a single [PortPrototype](#).] ([RS\\_SWCT\\_02060](#))

As indicated by Figure 7.44, there are two potentially competing [SwcServiceDependencies](#) that could be taken to contribute their `shortName` for filling in the suffix of the `DataServices_{Data}`.

In this case, it is actually necessary to settle the “over-supply of `shortNames`” by regulation of the AUTOSAR standard. [\[TPS\\_SWCT\\_01691\]](#) has been created for this purpose.

Another realistic example where [\[TPS\\_SWCT\\_01689\]](#) applies is an [AtomicSwComponentType](#) that exposes a [PPortPrototype](#) typed by a [SenderReceiverInterface](#) and the `dataElement(s)` within the [PPortPrototype](#) are both accessed as diagnostic values (see chapter 13.8.4.3) **and** are used to send mode requests to the BswM (see chapter 13.6.4).

Note that in this case a regulation regarding the `shortNames` of the affected `SwcServiceDependency` is not required because the applicable `SenderReceiverInterface` is not standardized and does not require the assignment of a name suffix from the existing model.

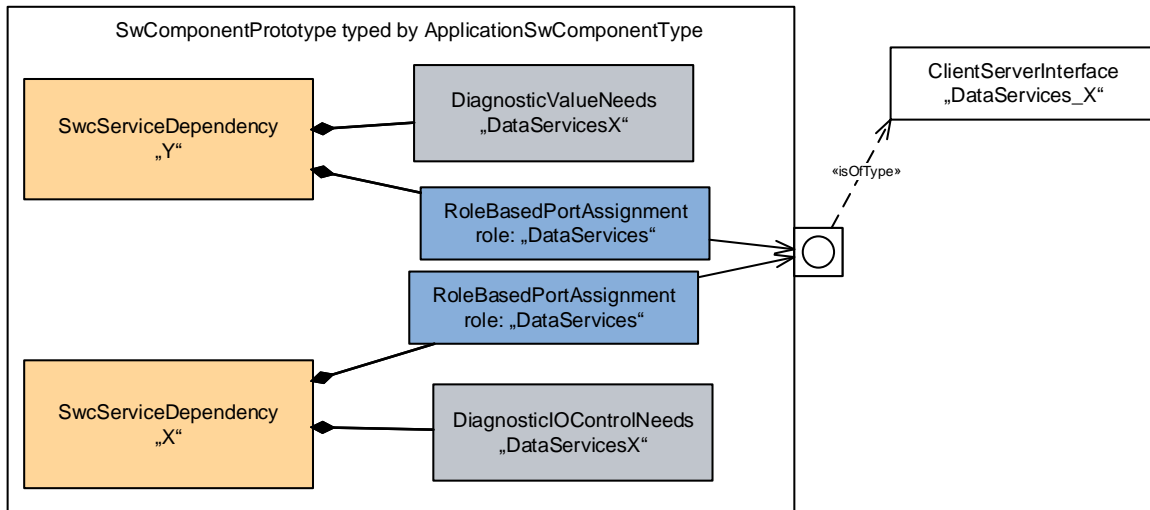


Figure 7.44: Two `SwcServiceDependency`s referencing one `PortPrototype`

**[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`. For this purpose the `VendorSpecificServiceNeeds` is available.] ([RS\\_SWCT\\_02060](#))

**[TPS\_SWCT\_01833] Semantics of `ServiceDependency.diagnosticRelevance`** [The attribute `ServiceDependency.diagnosticRelevance` can be used to indicate a diagnostic relevance of the `ServiceDependency`, especially for `ServiceDependencies` that are not related to diagnostics, but which are still required for the integration of the diagnostic stack.] ([RS\\_SWCT\\_02060](#))

One example for the usage of `ServiceDependency.diagnosticRelevance` would be the communication between the application software and the BswM, i.e. in the context of a `SwcServiceDependency` that aggregates a `BswMgrNeeds`. This communication may be relevant for the configuration of so-called "mode conditions" in the Dcm.

Please note that the modeling of attribute is very general and can be applied to every foreseeable service configuration. However, the only know use case that is actually relevant is the mentioned communication between application software and the BswM.

In order to not open the door for the uncontrolled usage of uncharted territory (that may nevertheless later be explored, use case by use case), restrictions apply for the usage of `ServiceDependency.diagnosticRelevance`, as documented in [[constr\\_10032](#)].

[constr\_10032] **Restrictions for the usage of `ServiceDependency.diagnosticRelevance`** [The attribute `ServiceDependency.diagnosticRelevance` shall only be used for a `SwcServiceDependency` that aggregates a `BswMgrNeeds` **at the time when the RTE is generated.**]([\(\)](#))

## 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 `VariationPointProxys`, 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 `postBuild-ValueAccess` shall yield an integer value conform with the `implementation-DataType`.

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 `Platform Data 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.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

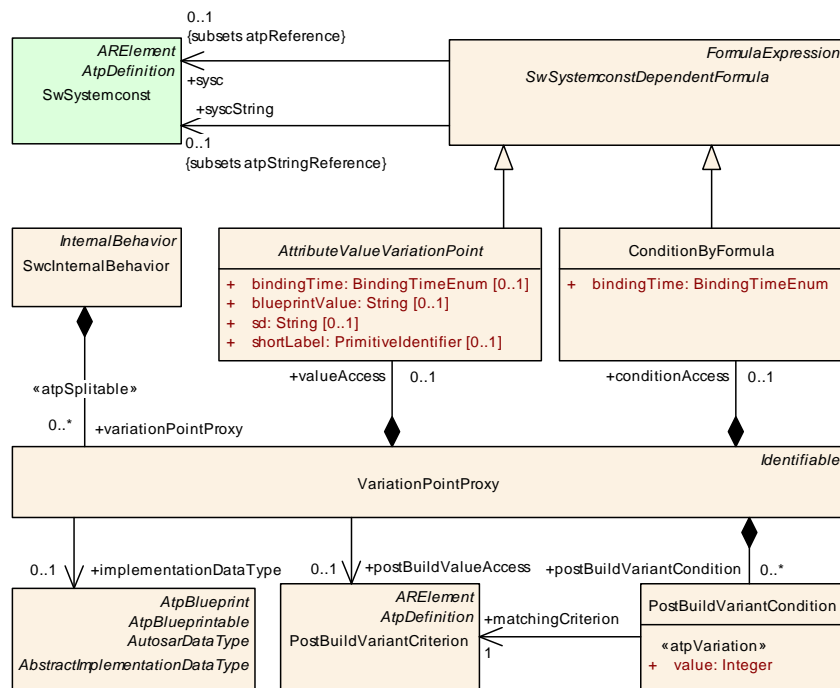
**[constr\_1389] Restriction regarding the value of category of VariationPointProxy.implementationDataType** [`VariationPointProxy.implementation-DataType` shall **not** be of `category STRUCTURE, ARRAY, UNION, FUNCTION_REFERENCE, and DATA_REFERENCE`.

The `VariationPointProxy.implementationDataType` shall be of category `VALUE` or `TYPE_REFERENCE` that, after all references are resolved, yields an `ImplementationDataType` of category `VALUE`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[TPS\_SWCT\_01372] `bindingTime = preCompileTime`** [In case of `bindingTime = preCompileTime` the RTE provides macro definitions that can be used for pre-processor 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 Pre-processor and to provide the resulting values to the software-component's implementation.] ([RS\\_SWCT\\_03100](#))



**Figure 7.45: VariationPointProxy**

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 [\[2\]](#).

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 7.64 or [`constr_1253`], respectively).

<b>Class</b>	<b>VariationPointProxy</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::VariantHandling			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
conditionAccess	<a href="#">ConditionByFormula</a>	0..1	aggr	This condition acts as Binding Function for the Variation Point.
implementationDataType	<a href="#">AbstractImplementationDataType</a>	0..1	ref	This association to ImplementationDataType shall be taken as an implementation hint by the RTE generator.
postBuildValueAccess	<a href="#">PostBuildVariantCriterion</a>	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	<a href="#">PostBuildVariantCondition</a>	*	aggr	This represents that applicable PostBuildVariantCondition in the context of aVariationPointProxy.
valueAccess	<a href="#">AttributeValueVariationPoint</a>	0..1	aggr	This value acts as Binding Function for the VariationPoint.

**Table 7.63: VariationPointProxy**

**[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 7.64.

For clarification, the multiplicities of attributes of meta-class [VariationPointProxy](#) that are **not** explicitly mentioned in a given row of table 7.64 shall be interpreted as [0].

This rule shall be applied **at the time when the contract phase generation is executed.**]()

BindingTime	category	Allowed Attribute Multiplicity
<i>PreBuild</i>	VALUE	<a href="#">valueAccess</a> [1]
	CONDITION	<a href="#">conditionAccess</a> [1]
<i>PostBuild</i>	VALUE	<a href="#">postBuildValueAccess</a> [1], <a href="#">implementationDataType</a> [1]
	CONDITION	<a href="#">postBuildVariantCondition</a> [1..*], <a href="#">conditionAccess</a> [0..1]

**Table 7.64: Supported usage of [VariationPointProxy](#)**

<b>Class</b>	<<atpMixedString>> <b>ConditionByFormula</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	<p>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.</p> <p>The result of the expression is interpreted as a condition.</p> <ul style="list-style-type: none"> <li>• "0" represents "false";</li> <li>• a value other than zero is considered "true"</li> </ul>			
<b>Base</b>	<i>ARObject</i> , <i>FormulaExpression</i> , <i>SwSystemconstDependentFormula</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
bindingTime	<a href="#">BindingTimeEnum</a>	1	attr	<p>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.</p> <p><b>Tags:</b>xml.attribute=true</p>

**Table 7.65: ConditionByFormula**

<b>Class</b>	<<atpMixedString>> <b>AttributeValueVariationPoint</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling::AttributeValueVariationPoints			
<b>Note</b>	<p>This class represents the ability to derive the value of the Attribute from a system constant (by Sw SystemconstDependentFormula). It also provides a bindingTime.</p>			
<b>Base</b>	<i>ARObject</i> , <i>FormulaExpression</i> , <i>SwSystemconstDependentFormula</i>			
<b>Subclasses</b>	<i>AbstractEnumerationValueVariationPoint</i> , <i>AbstractNumericalVariationPoint</i> , <i>BooleanValueVariationPoint</i> , <i>FloatValueVariationPoint</i> , <i>IntegerValueVariationPoint</i> , <i>PositiveIntegerValueVariationPoint</i> , <i>TimeValueVariationPoint</i> , <i>UnlimitedIntegerValueVariationPoint</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
bindingTime	<a href="#">BindingTimeEnum</a>	0..1	attr	<p>This is the binding time in which the attribute value needs to be bound.</p> <p>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.</p> <p><b>Tags:</b>xml.attribute=true</p>
blueprintValue	<a href="#">String</a>	0..1	attr	<p>This represents a description that documents how the value shall be defined when deriving objects from the blueprint.</p> <p><b>Tags:</b>xml.attribute=true</p>
sd	<a href="#">String</a>	0..1	attr	<p>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.</p> <p><b>Tags:</b>xml.attribute=true</p>
shortLabel	<a href="#">PrimitivIdentifier</a>	0..1	attr	<p>This allows to identify the variation point. It is also intended to allow RTE support for CompileTime Variation points.</p> <p><b>Tags:</b>xml.attribute=true</p>

**Table 7.66: AttributeValueVariationPoint**

<b>Class</b>	<b>PostBuildVariantCriterion</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	This class specifies one particular PostBuildVariantSelector. <b>Tags:</b> atp.recommendedPackage=PostBuildVariantCriteria			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpDefinition</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuMethod	<a href="#">CompuMethod</a>	1	ref	The compuMethod specifies the possible values for the variant criterion serving as an enumerator.

**Table 7.67: PostBuildVariantCriterion**

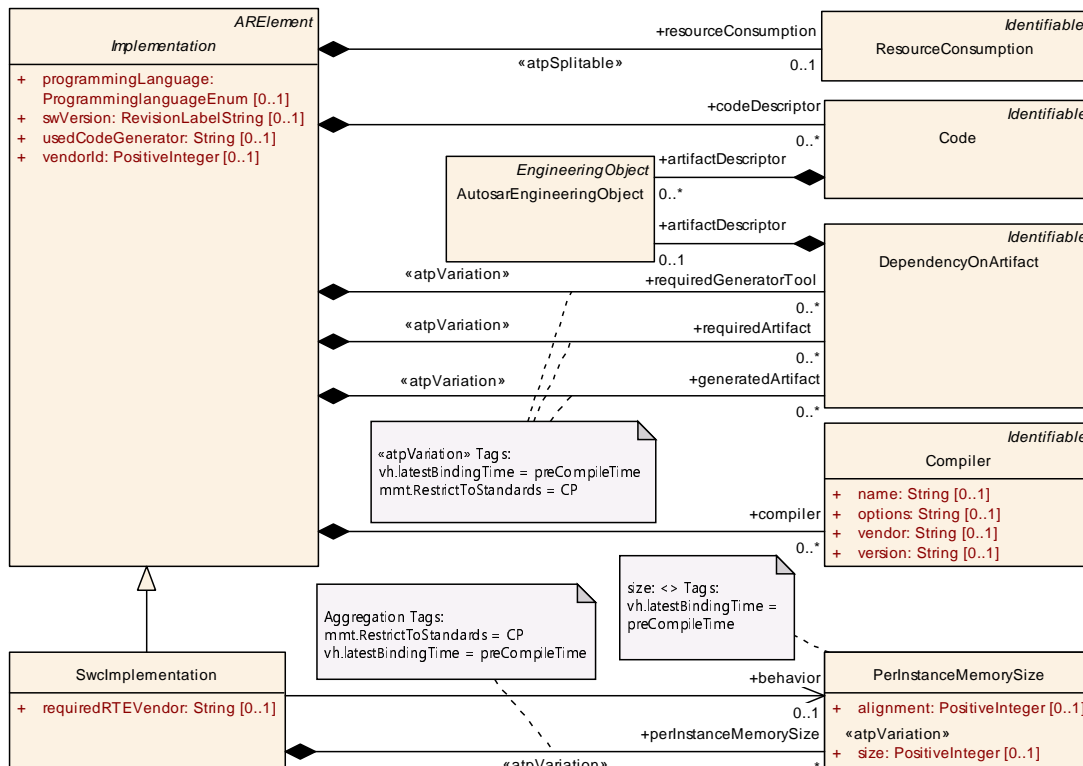
<b>Class</b>	<b>PostBuildVariantCondition</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	This class specifies the value which shall be assigned to a particular variant criterion in order to bind the variation point. If multiple criterion/value pairs are specified, they shall all match to bind the variation point. In other words binding can be represented by (criterion1 == value1) && (condition2 == value2) ...			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
matching Criterion	<a href="#">PostBuildVariantCriterion</a>	1	ref	This is the criterion which needs to match the value in order to make the PostbuildVariantCondition to be true.
value	Integer	1	attr	This is the particular value of the post-build variant criterion. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 7.68: PostBuildVariantCondition**



## 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 [6].



**Figure 8.1: Implementation part specific to the Software Component Template**

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.

<b>Class</b>	<i>Implementation</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Implementation			
<b>Note</b>	Description of an implementation a single software component or module.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">BswImplementation</a> , <a href="#">SwcImplementation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	Implementation (abstract)			
buildActionManifest	BuildActionManifest	0..1	ref	A manifest specifying the intended build actions for the software delivered with this implementation. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=codeGenerationTime
codeDescriptor	<a href="#">Code</a>	*	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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
hwElement	<a href="#">HwElement</a>	*	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	<a href="#">McSupportData</a>	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. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=mcSupport
programmingLanguage	ProgrammingLanguageEnum	0..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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
requiredGeneratorTool	DependencyOnArtifact	*	aggr	Relates this Implementation to a generator tool in order to generate additional artifacts during integration. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
resourceConsumption	ResourceConsumption	0..1	aggr	All static and dynamic resources for each implementation are described within the ResourceConsumption class. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=resourceConsumption.shortName
swcBswMapping	<a href="#">SwcBswMapping</a>	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 Bsw Implementation or for both.
swVersion	RevisionLabelString	0..1	attr	Software version of this implementation. The numbering contains three levels (like major, minor, patch), its values are vendor specific.
usedCodeGenerator	<a href="#">String</a>	0..1	attr	Optional: code generator used.





<b>Class</b>	<b>Implementation</b> (abstract)			
vendorId	PositiveInteger	0..1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list

**Table 8.1: Implementation**

**[constr\_1966] Existence of attribute `Implementation.swVersion`** [For each `Implementation`, attribute `swVersion` shall exist **at the time when the RTE is generated.**]()

**[constr\_1967] Existence of attribute `Implementation.vendorId`** [For each `Implementation`, attribute `vendorId` shall exist **at the time when the RTE is generated.**]()

**[constr\_1968] Existence of attribute `Implementation.codeDescriptor`** [For each `Implementation`, at least one aggregation of `Code` in the role `codeDescriptor` shall exist **at the time when the RTE is generated.**]()

<b>Class</b>	<b>Code</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Implementation			
<b>Note</b>	A generic code descriptor. The type of the code (source or object) is defined via the category attribute of the associated engineering object.			
<b>Base</b>	<i>ARObject, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
artifactDescriptor	AutosarEngineeringObject	*	aggr	Refers to the artifact belonging to this code descriptor.
callbackHeader	<a href="#">ServiceNeeds</a>	*	ref	The association callbackHeader describes in which header files the function declarations of callback functions are provided to a service module. With this information the service module can include the appropriate header files in its configuration files.

**Table 8.2: Code**

<b>Class</b>	<b>SwcImplementation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcImplementation			
<b>Note</b>	This meta-class represents a specialization of the general Implementation meta-class with respect to the usage in application software. <b>Tags:</b> atp.recommendedPackage=SwcImplementations			
<b>Base</b>	<i>ARElement, ARObject, CollectableElement, Identifiable, Implementation, MultilanguageReferrable, PackageableElement, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
behavior	<a href="#">SwcInternalBehavior</a>	0..1	ref	The internal behavior implemented by this Implementation.





Class	SwcImplementation			
perInstanceMemorySize	<a href="#">PerInstanceMemorySize</a>	*	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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
requiredRTEVendor	<a href="#">String</a>	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 8.3: SwcImplementation**

**[constr\_1969] Existence of attribute [SwcImplementation.behavior](#)** [For each [SwcImplementation](#), attribute [behavior](#) shall exist **at the time when the RTE is generated.**]()

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	<a href="#">ARObject</a>			
Attribute	Type	Mult.	Kind	Note
alignment	PositiveInteger	0..1	attr	Required alignment (1,2,4,...) of the referenced PerInstanceMemory. Unit: byte.
perInstanceMemory	<a href="#">PerInstanceMemory</a>	0..1	ref	This represents the referenced PerInstanceMemory.
size	PositiveInteger	0..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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 8.4: PerInstanceMemorySize**

**[constr\_1970] Existence of attribute [PerInstanceMemorySize.alignment](#)** [For each [PerInstanceMemorySize](#), attribute [alignment](#) shall exist **at the time when the RTE is generated.**]()

**[constr\_1971] Existence of attribute [PerInstanceMemorySize.perInstanceMemory](#)** [For each [PerInstanceMemorySize](#), the reference to [PerInstanceMemory](#) in the role [perInstanceMemory](#) shall exist **at the time when the RTE is generated.**]()

**[constr\_1972] Existence of attribute `PerInstanceMemorySize.size`** [For each `PerInstanceMemorySize`, attribute `size` shall exist **at the time when the RTE is generated.**]()

## 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`<sup>1</sup> 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` at the time when the RTE is generated.] ()

### 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](#)

---

<sup>1</sup>See also AUTOSAR Glossary for "Standardized AUTOSAR Interface".

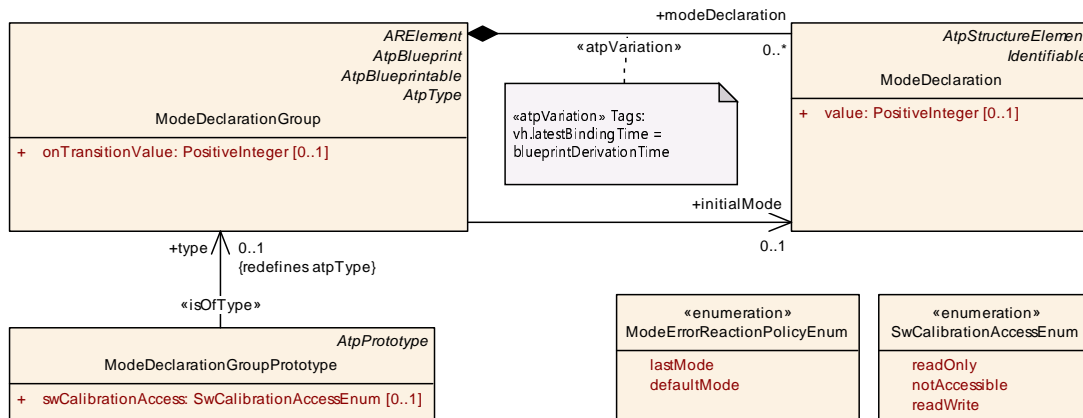


Figure 9.1: ModeDeclaration

The class `ModeDeclarationGroup` has been introduced to support the grouping of modes and (on M1 level) to provide predefined sets of modes that could be standardized and re-used. The set of modes eventually defines a flat (i.e. no hierarchical states) state-machine where only one mode can be active at a given point in time.

Again, please note that the actual definition of modes and their relationship is not in the responsibility of this document. In other words: the definition of modes represents M1 artifacts whereas this document is limited to describing M2 model elements.

Both `ModeDeclaration` and `ModeDeclarationGroup` own attributes that facilitate the generation of C source code from the formal definition.

**[TPS\_SWCT\_01008] Definition of positive integer values that are directly taken over by the RTE generator for creating the programmatic representations of the ModeDeclaration** [The attributes `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue` allow for the definition of positive integer values that are directly taken over by the RTE generator for creating the programmatic representations of the `ModeDeclaration` and `ModeDeclarationGroup` in the source code.] (*RS\_SWCT\_03200*)

**[constr\_1399] Standardized values of ModeDeclarationGroup.category** [The AUTOSAR standard defines the following values of the attribute `ModeDeclarationGroup.category` with a standardized meaning:

- EXPLICIT\_ORDER
- ALPHABETIC\_ORDER

[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`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

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`** [The attributes `ModeDeclarationGroup.onTransitionValue` and `ModeDeclaration.value` (for each `ModeDeclaration`) shall be set if the `category` of a `ModeDeclarationGroup` is set to `EXPLICIT_ORDER`.

This rule shall be imposed **at the time when the contract phase generation is executed.**]()

**[constr\_1299] Existence of attributes if `category` of a `ModeDeclarationGroup` is set to other than `EXPLICIT_ORDER`** [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` **at the time when the contract phase generation is executed.**]()

**[constr\_1181] Numerical values used in `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue`** [The numerical values used to define the `value` attributes and the `onTransitionValue` attribute of a `ModeDeclarationGroup` shall not overlap **at the time when the contract phase generation is executed.**]()

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** [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.](*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 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`** [In order to support a clear separation between the two possible ways to influence the



definition of the programmatic representation of `ModeDeclarations` two `category`s 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)

<b>Class</b>	<b>ModeDeclaration</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
<b>Base</b>	<i>ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this Mode Declaration.

**Table 9.1: ModeDeclaration**

<b>Class</b>	<b>ModeDeclarationGroup</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	A collection of Mode Declarations. Also, the initial mode is explicitly identified. <b>Tags:</b> atp.recommendedPackage=ModeDeclarationGroups			
<b>Base</b>	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initialMode	<a href="#">ModeDeclaration</a>	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
mode Declaration	<a href="#">ModeDeclaration</a>	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime
modeManager ErrorBehavior	<a href="#">ModeErrorBehavior</a>	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	<a href="#">ModeTransition</a>	*	aggr	This represents the available ModeTransitions of the ModeDeclarationGroup





Class	ModeDeclarationGroup			
modeUserError Behavior	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).
onTransition Value	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**

**[constr\_1973] Existence of attribute ModeDeclarationGroup.initialMode**  
 [For each ModeDeclarationGroup, the reference to ModeDeclaration in the role initialMode shall exist **at the time when the contract phase generation is executed.**]()

**[constr\_1974] Existence of attribute ModeDeclarationGroup.modeDeclaration**  
 [For each ModeDeclarationGroup, at least one ModeDeclaration shall be aggregated in the role modeDeclaration **at the time when the contract phase generation is executed.**]()

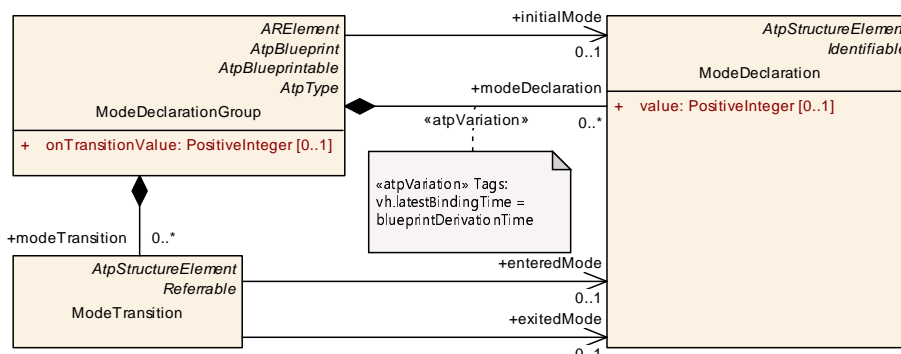
**[TPS\_SWCT\_01450] Semantics of a ModeTransition** [In addition to the ability to specify ModeDeclarations within a ModeDeclarationGroup it is also feasible to define possible transitions between ModeDeclarations within the given ModeDeclarationGroup. This can be done by means of aggregation ModeTransition at ModeDeclarationGroup in the role modeTransition.] (RS\_SWCT\_03200)

More details are explained in Figure 9.2.

**[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 **at the time when the RTE is generated 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.

<b>Class</b>	<b>ModeTransition</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	This meta-class represents the ability to describe possible ModeTransitions in the context of a Mode DeclarationGroup.			
<b>Base</b>	<i>ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
enteredMode	<code>ModeDeclaration</code>	0..1	ref	This represents the entered model of the ModeTransition.
exitedMode	<code>ModeDeclaration</code>	0..1	ref	This represents the exited mode of the ModeTransition

**Table 9.3: ModeTransition**

**[constr\_1975] Existence of attribute ModeTransition.enteredMode** [For each `ModeTransition`, the reference to `ModeDeclaration` in the role `enteredMode` shall exist **at the time when the RTE is generated**.]()

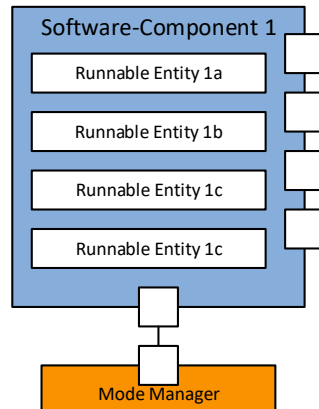
**[constr\_1976] Existence of attribute ModeTransition.exitedMode** [For each `ModeTransition`, the reference to `ModeDeclaration` in the role `exitedMode` shall exist **at the time when the RTE is generated**.]()

## 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](#))

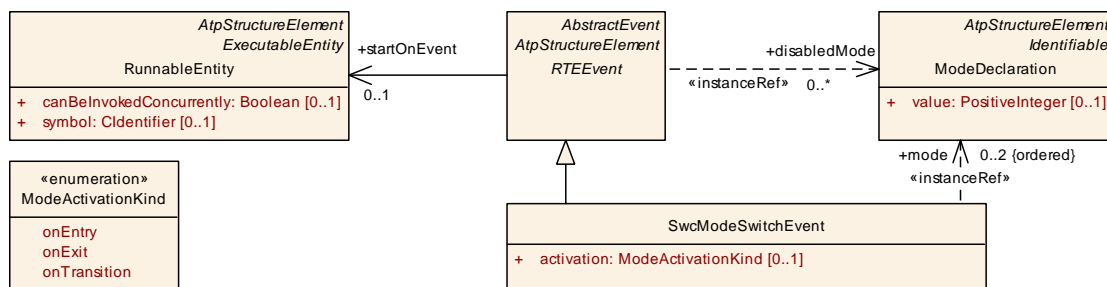
Such a mode dependent software-component is shown in Figure 9.3.

**[TPS\_SWCT\_01077] Configure the response to mode changes** [Since the behavior of `AtomicSwComponentTypes` is mainly determined by the `RunnableEntitys` contained in the `SwcInternalBehavior`, it is necessary to configure the response to mode changes on the level of `RunnableEntitys`.]([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.] ([RS\\_SWCT\\_03110](#))

Both mechanisms are visible in Figure 9.4.

**[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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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 **at the time when the RTE is generated** 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.

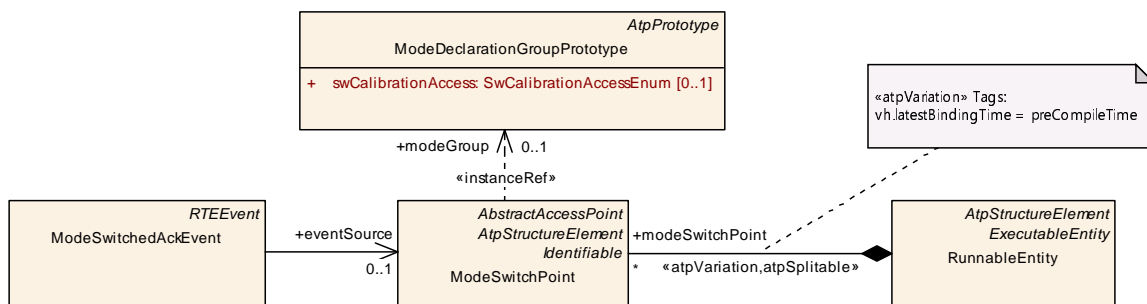
That is, RTEEvents without an association in the role disabledMode are processed regularly according to their definition.

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.](RS\_SWCT\_03110)

For more information, please refer to Figure 9.5.



**Figure 9.5: ModeSwitchPoint**

**[constr\_1778] Value of attribute modeSwitchPoint.returnValueProvision** [All RunnableEntity.modeSwitchPoint that refer to the same modeGroup shall define the identical value of attribute returnValueProvision **at the time when the contract phase generation is executed.**]()

Rationale for the existence of [constr\_1778]: different `RunnableEntity`s could aggregate `ModeSwitchPoints` with a different configuration of attribute `returnValueProvision`.

However, in such a case it would not be possible to generate the corresponding RTE API because the API exists once per software-component and it is therefore indispensable to have all affected `ModeSwitchPoints` agree on the configuration of attribute `returnValueProvision`.

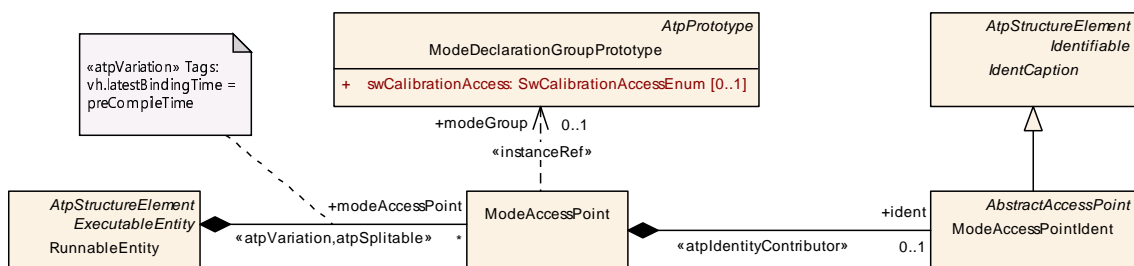
This relation is exemplarily (for the case of `dataSendPoint.returnValueProvision`) sketched in Figure 7.21.

<b>Class</b>	<b>ModeSwitchPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ModeDeclarationGroup			
<b>Note</b>	A <code>ModeSwitchPoint</code> is required by a <code>RunnableEntity</code> owned a Mode Manager. Its semantics implies the ability to initiate a mode switch.			
<b>Base</b>	<i>ARObject</i> , <i>AbstractAccessPoint</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
modeGroup	<a href="#">ModeDeclarationGroup Prototype</a>	0..1	iref	The mode declaration group that is switched by this runnable.  <b>InstanceRef implemented by:</b> <code>PModeGroupInAtomicSwcInstanceRef</code>

**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`. 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](#))



**Figure 9.6: ModeAccessPoint**

[constr\_4012] **Timeout of ModeSwitchedAckEvent** [The timeout value of a `WaitPoint` associated with a `ModeSwitchedAckEvent` shall be equal to the corresponding `ModeSwitchedAckRequest.timeout` **at the time when the RTE is generated.**]([RS\\_SWCT\\_03110](#))

**[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, a [RunnableEntity](#) that requires read-access to the [ModeDeclarationGroupPrototype](#)'s current mode has to define a [ModeAccessPoint](#).] ([RS\\_SWCT\\_03110](#))

<b>Class</b>	<b>ModeAccessPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ModeDeclarationGroup			
<b>Note</b>	A <a href="#">ModeAccessPoint</a> is required by a <a href="#">RunnableEntity</a> owned by a <i>Mode Manager</i> or <i>Mode User</i> . Its semantics implies the ability to access the current mode (provided by the RTE) of a <a href="#">ModeDeclarationGroupPrototype</a> 's <a href="#">ModeDeclarationGroup</a> .			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
ident	<a href="#">ModeAccessPointIdent</a>	0..1	aggr	The aggregation in the role <code>ident</code> provides the ability to make the <a href="#">ModeAccessPoint</a> identifiable.  From the semantical point of view, the <a href="#">ModeAccessPoint</a> is considered a first-class <a href="#">Identifiable</a> and therefore the aggregation in the role <code>ident</code> shall always exist (until it may be possible to let <a href="#">ModeAccessPoint</a> directly inherit from <a href="#">Identifiable</a> ).  <b>Stereotypes:</b> <code>atpIdentityContributor</code> <b>Tags:</b> <code>xml.sequenceOffset=-100</code>
modeGroup	<a href="#">ModeDeclarationGroupPrototype</a>	0..1	iref	The mode declaration group that is accessed by this runnable.  <b>Tags:</b> <code>xml.typeElement=true</code> <b>InstanceRef implemented by:</b> <a href="#">ModeGroupInAtomicSwcInstanceRef</a>

**Table 9.5: 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 reading 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` **at the time when the RTE is generated**.] ()

If [\[constr\\_1098\]](#) would not apply, it might happen that a [RunnableEntity](#) would be triggered by a [SwcModeSwitchEvent](#) and, at 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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

Please note that the modeling of the `ModeErrorBehavior` is depicted in Figure 7.10.

**[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*)

<b>Class</b>	<b>ModeErrorBehavior</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	This represents the ability to define the error behavior in the context of mode handling.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
defaultMode	<code>ModeDeclaration</code>	0..1	ref	This represents the <code>ModeDeclaration</code> that is considered the error mode in the context of the enclosing <code>ModeDeclarationGroup</code> .
errorReaction Policy	<code>ModeErrorReaction PolicyEnum</code>	0..1	attr	This represents the ability to define the policy in terms of which default model shall apply in case an error occurs.

**Table 9.6: ModeErrorBehavior**

**[constr\_1977] Existence of attribute `ModeErrorBehavior.errorReactionPolicy`** [For each `ModeErrorBehavior`, the attribute `errorReactionPolicy` shall exist **at the time when the RTE is generated.**]()

<b>Enumeration</b>	<b>ModeErrorReactionPolicyEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration
<b>Note</b>	This represents the ability to specify the reaction on a mode error.
<b>Literal</b>	<b>Description</b>
defaultMode	This represents the ability to switch to the defaultMode in case of a mode error. <b>Tags:</b> atp.EnumerationLiteralIndex=0
lastMode	This represents the ability to keep the last mode in case of a mode error. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 9.7: 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*)

<b>Class</b>	<b>SwcModeManagerErrorEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	This event is raised when an error occurred during the handling of the referenced ModeDeclarationGroup Prototype.			
<b>Base</b>	<i>ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, Multilanguage Referrable, RTEEvent, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
modeGroup	<code>ModeDeclarationGroup Prototype</code>	0..1	iref	This represents the ModeDeclarationGroupPrototype for which this SwcModeManagerErrorEvent is raised in case of an error. <b>InstanceRef implemented by:</b> <code>PModeGroupInAtomic SwcInstanceRef</code>

**Table 9.8: SwcModeManagerErrorEvent**

**[constr\_1978] Existence of attribute `SwcModeManagerErrorEvent.modeGroup`** [For each `SwcModeManagerErrorEvent`, the instance reference to `ModeDeclaration` in the role `modeGroup` shall exist **at the time when the RTE is generated.**]()

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**.

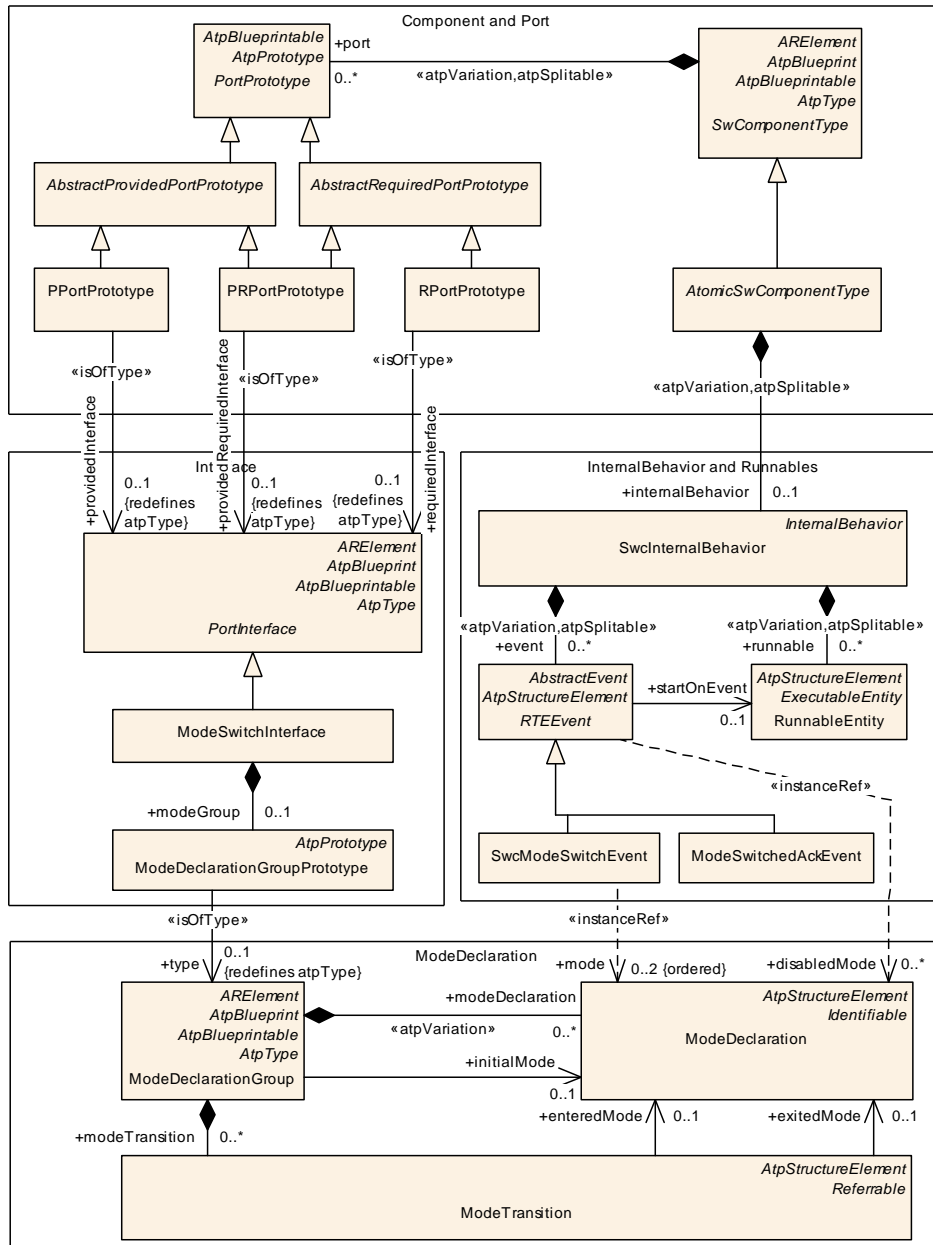
Please note that the statements made in [\[TPS\\_SWCT\\_01541\]](#) is further underlined by [\[SWS\\_Rte\\_06795\]](#) and the statement made by [\[TPS\\_SWCT\\_01542\]](#) is further underlined by [\[SWS\\_Rte\\_06795\]](#).

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 [9.7](#) provides an overview of all meta-model elements that have a direct relationship to the meta-classes involved in the modeling 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 9.7: Summary meta-model excerpt related to modes**

## 10 ECU Abstraction and Complex Drivers

### 10.1 Introduction

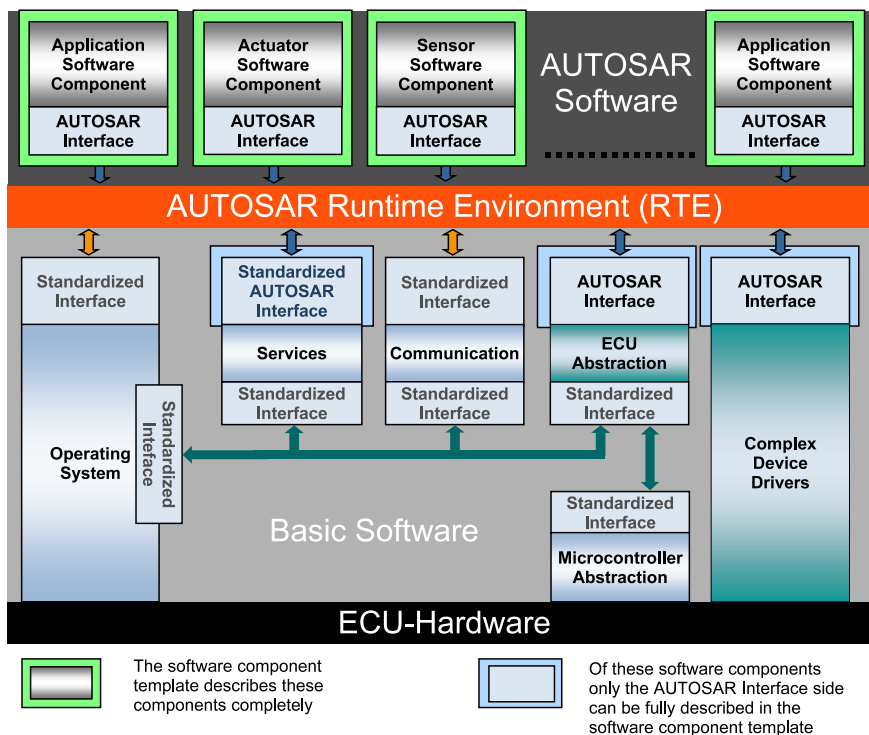
During the design of embedded systems there is one crucial point where the hardware and software have to be related. In AUTOSAR the ECU Resource Template describes the provided hardware resources.

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 [6]. 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**

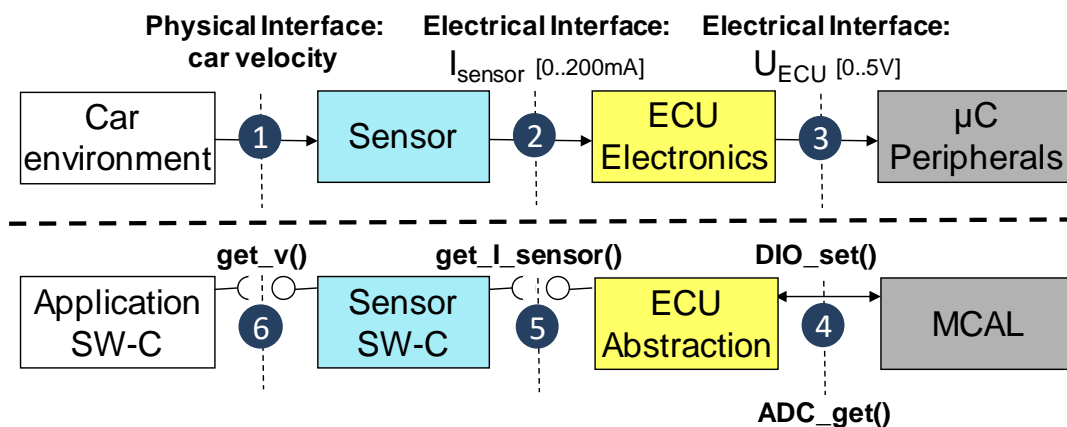
The signal<sup>1</sup> flow from a hardware to software and vice versa will be described in the following sections.

A sensor<sup>2</sup> 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 device is also responsible for the conversion of the electrical signal into a micro-controller compatible form (3), usually a voltage.

After the electrical signal has been enhanced and converted it will be captured by the micro-controller. 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.



**Figure 10.2: Interfaces between hardware and software**

The interaction shown in Figure 10.2 shows an exemplary scenario. The usage of `ClientServerInterface` is just an example for the interaction pattern.

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 micro-controller 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.

<sup>1</sup>The 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.

<sup>2</sup>For the sake of simplicity this discussion is limited to the sensor aspects. Nevertheless, the same applies also for actuators.

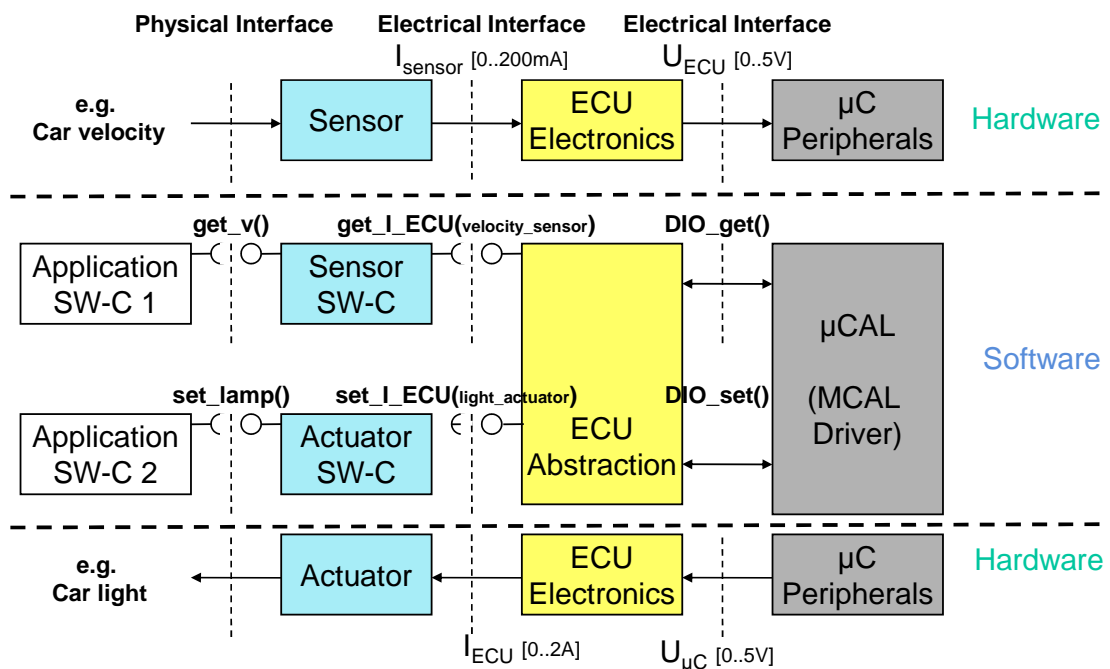
So if an electrical current is the input to the micro-controller peripheral, the MCAL will deliver a data value that represents this current. As the ECU Electronics has enhanced and converted the electrical signal prior to the micro-controller, the corresponding software entity is reversing this conversion. 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**

The interaction shown in Figure 10.3 shows an exemplary scenario. The usage of `ClientServerInterface` is just an example for the interaction pattern.

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 micro-controller Abstraction and the ECU Abstraction is a Standardized Interface (see AUTOSAR Glossary [34]). 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 `SensorActuatorSwComponentType` 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.

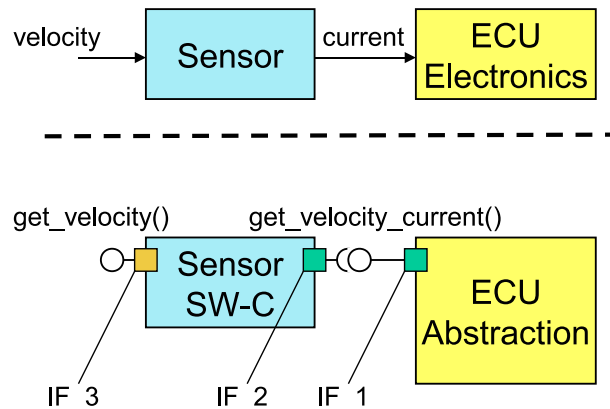
In the case of the sensors and actuators the interface is gathered in the ECU Abstraction. For each sensor and actuator there is one AUTOSAR `PortPrototype` that represents the AUTOSAR Signal that is delivered by the sensor or the AUTOSAR Signal that is consumed by the actuator. This relationship is depicted in Figure 10.4.

The interaction shown in Figure 10.4 shows an exemplary scenario. The usage of `ClientServerInterface` is just an example for the interaction pattern.

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.





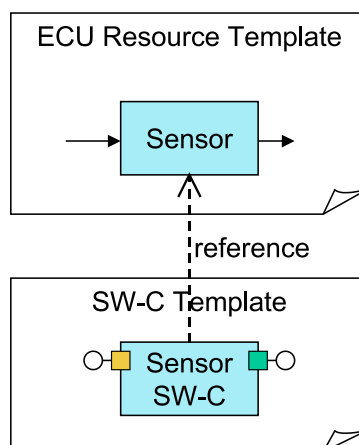
**Figure 10.4: Interfaces of signals in software**

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 [5] 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](#))

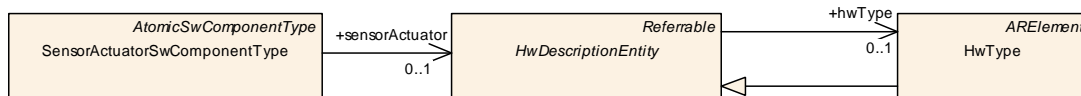


**Figure 10.5: Shipment of a sensor**

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 **at the time when the RTE is generated.**]()

Figure 10.6 depicts the reference of `SensorActuatorSwComponentType` designed as a specialization of an `AtomicSwComponentType` with an additional reference to a `HwType`.



**Figure 10.6: Sensor/actuator to Hardware Relationship**

**[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.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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*)

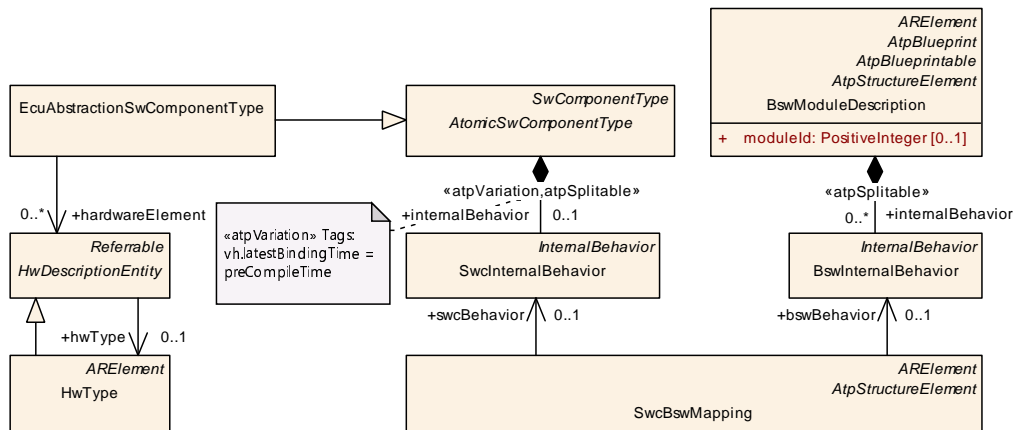
In case the sensor/actuator hardware is accessed via bus communication, e.g. is located on a LIN slave, no such mapping constraints apply (note that this is not handled via the IO hardware abstraction layer).

<b>Class</b>	<b>SensorActuatorSwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	The <code>SensorActuatorSwComponentType</code> introduces the possibility to link from the software representation of a sensor/actuator to its hardware description provided by the ECU Resource Template. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<i>ARElement, ARObjct, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
sensorActuator	<code>HwDescriptionEntity</code>	0..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`.]()



**Figure 10.7: 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 `EcuAbstractionSwComponentTypes` on M1.]()

<b>Class</b>	<b>EcuAbstractionSwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	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. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	ARElement, ARObject, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
hardware Element	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 [6].]()

## 10.6 Complex Driver

[TPS\_SWCT\_01393] **Complex Driver** [A `Complex Driver` implements complex sensor evaluation and actuator control with direct access to the micro-controller using specific interrupts and/or complex micro-controller 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 [ComplexDeviceDriverSwComponentTypes](#).]()

<b>Class</b>	<b>ComplexDeviceDriverSwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	The <a href="#">ComplexDeviceDriverSwComponentType</a> is a special <a href="#">AtomicSwComponentType</a> that has direct access to hardware on an ECU and which is therefore linked to a specific ECU or specific hardware. The <a href="#">ComplexDeviceDriverSwComponentType</a> introduces the possibility to link from the software representation to its hardware description provided by the ECU Resource Template. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtomicSwComponentType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
hardware Element	<a href="#">HwDescriptionEntity</a>	*	ref	Reference from the <a href="#">ComplexDeviceDriverSwComponentType</a> to the description of the used <a href="#">HwElements</a> .

**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 [6].()

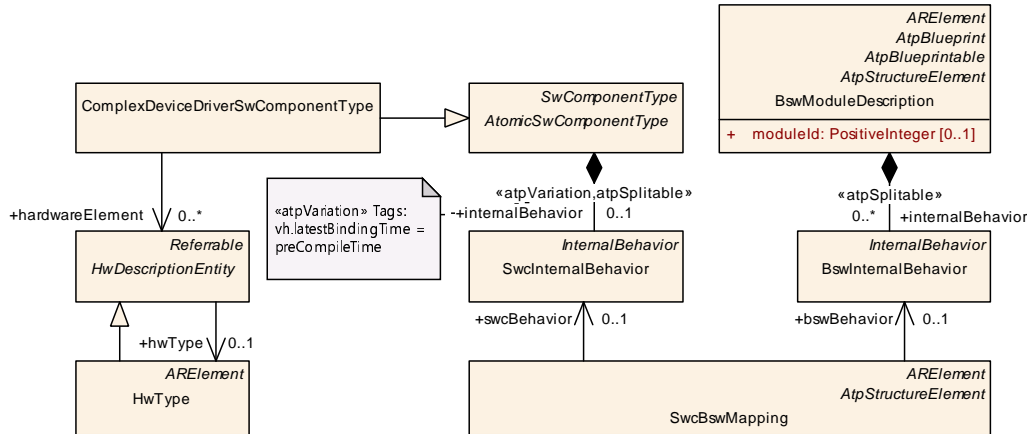


Figure 10.8: `ComplexDeviceDriverSwComponentType`

[constr\_1979] Existence of the reference `SwcBswMapping.bswBehavior` [For each `SwcBswMapping`, the reference to `BswInternalBehavior` in the role `bswBehavior` shall exist at the time when the RTE is generated.]()

[constr\_1980] Existence of the reference `SwcBswMapping.swcBehavior` [For each `SwcBswMapping`, the reference to `BswInternalBehavior` in the role `swcBehavior` shall exist at the time when the RTE is generated.]()

<b>Class</b>	<b>BswInternalBehavior</b>			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
<b>Note</b>	Specifies the behavior of a BSW module or a BSW cluster w.r.t. the code entities visible by the BSW Scheduler. It is possible to have several different <code>BswInternalBehaviors</code> referring to the same <code>BswModuleDescription</code> .			
<b>Base</b>	<code>ARObject</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>InternalBehavior</code> , <code>Multilanguage</code> , <code>Referrable</code> , <code>Referrable</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arTypedPer Instance Memory	<code>VariableDataPrototype</code>	*	aggr	Defines an AUTOSAR typed memory-block that needs to be available for each instance of the Basic Software Module. The aggregation of <code>arTypedPerInstanceMemory</code> is subject to variability with the purpose to support variability in the Basic Software Module's implementations. Typically different algorithms in the implementation are requiring different number of memory objects.  <b>Stereotypes:</b> <code>atpSplittable</code> ; <code>atpVariation</code> <b>Tags:</b> <code>atp.Splitkey=arTypedPerInstanceMemory.shortName</code> , <code>arTypedPerInstanceMemory.variationPoint.shortLabel</code> , <code>vh.latestBindingTime=preCompileTime</code>





Class	BswInternalBehavior			
bswPerInstanceMemoryPolicy	BswPerInstanceMemoryPolicy	*	aggr	<p>Policy for a arTypedPerInstanceMemory. The policy selects the options of the Schedule Manager API generation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=bswPerInstanceMemoryPolicy, bswPerInstanceMemoryPolicy.variationPoint.shortLabel            vh.latestBindingTime=preCompileTime</p>
clientPolicy	BswClientPolicy	*	aggr	<p>Policy for a requiredClientServerEntry. The policy selects the options of the Schedule Manager API generation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=clientPolicy, clientPolicy.variationPoint.shortLabel            vh.latestBindingTime=preCompileTime</p>
distinguishedPartition	BswDistinguishedPartition	*	aggr	<p>Indicates an abstract partition context in which the enclosing BswModuleEntity can be executed.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=distinguishedPartition.shortName, distinguishedPartition.variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=60</p>
entity	BswModuleEntity	*	aggr	<p>A code entity for which the behavior is described</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=entity.shortName, entity.variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=5</p>
event	BswEvent	*	aggr	<p>An event required by this module behavior.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=event.shortName, event.variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=10</p>
exclusiveAreaPolicy	BswExclusiveAreaPolicy	*	aggr	<p>Policy for an ExclusiveArea in this BswInternalBehavior. The policy selects the options of the Schedule Manager API generation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=exclusiveAreaPolicy, exclusiveAreaPolicy.variationPoint.shortLabel            vh.latestBindingTime=preCompileTime</p>
includedDataTypeSet	<a href="#">IncludedDataTypeSet</a>	*	aggr	<p>The includedDataTypeSet is used by a basic software module for its implementation.</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>atp.Splitkey=includedDataTypeSet</p>
includedModeDeclarationGroupSet	<a href="#">IncludedModeDeclarationGroupSet</a>	*	aggr	<p>This aggregation represents the included Mode DeclarationGroups</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>atp.Splitkey=includedModeDeclarationGroupSet</p>





Class	BswInternalBehavior			
internalTriggeringPoint	BswInternalTriggeringPoint	*	aggr	<p>An internal triggering point.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=internalTriggeringPoint.shortName, internalTriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=2</p>
internalTriggeringPointPolicy	BswInternalTriggeringPointPolicy	*	aggr	<p>Policy for an internalTriggeringPoint in this BswInternalBehavior.. The policy selects the options of the Schedule Manager API generation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=internalTriggeringPointPolicy, internalTriggeringPointPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
modeReceiverPolicy	BswModeReceiverPolicy	*	aggr	<p>Implementation policy for the reception of mode switches.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=modeReceiverPolicy, modeReceiverPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25</p>
modeSenderPolicy	BswModeSenderPolicy	*	aggr	<p>Implementation policy for providing a mode group.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=modeSenderPolicy, modeSenderPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20</p>
parameterPolicy	BswParameterPolicy	*	aggr	<p>Policy for a perInstanceParameter in this BswInternalBehavior. The policy selects the options of the Schedule Manager API generation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=parameterPolicy, parameterPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
perInstanceParameter	<a href="#">ParameterData Prototype</a>	*	aggr	<p>Describes a read only memory object containing characteristic value(s) needed by this BswInternalBehavior. The role name perInstanceParameter is chosen in analogy to the similar role in the context of SwcInternalBehavior.</p> <p>In contrast to constantMemory, this object is not allocated locally by the module's code, but by the BSW Scheduler and it is accessed from the BSW module via the BSW Scheduler API. The main use case is the support of software emulation of calibration data.</p> <p>The aggregation is subject to variability with the purpose to support implementation variants.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=perInstanceParameter.shortName, perInstanceParameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45</p>





Class	BswInternalBehavior			
receptionPolicy	BswDataReception Policy	*	aggr	Data reception policy for inter-partition and/or inter-core communication. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=receptionPolicy, receptionPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
releasedTrigger Policy	BswReleasedTrigger Policy	*	aggr	Policy for a releasedTrigger. The policy selects the options of the Schedule Manager API generation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=releasedTriggerPolicy, releasedTriggerPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
schedulerName Prefix	BswSchedulerName Prefix	*	aggr	Optional definition of one or more prefixes to be used for the BswScheduler. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=schedulerNamePrefix.shortName, schedulerNamePrefix.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
sendPolicy	BswDataSendPolicy	*	aggr	Policy for a providedData. The policy selects the options of the Schedule Manager API generation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=sendPolicy, sendPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
service Dependency	<a href="#">BswService Dependency</a>	*	aggr	Defines the requirements on AUTOSAR Services for a particular item.  The aggregation is subject to variability with the purpose to support the conditional existence of ServiceNeeds.  The aggregation is splitable in order to support that ServiceNeeds might be provided in later development steps. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=serviceDependency.ident.shortName, serviceDependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
triggerDirect Implementation	BswTriggerDirect Implementation	*	aggr	Specifies a trigger to be directly implemented via OS calls. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=triggerDirectImplementation, triggerDirectImplementation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=15
variationPoint Proxy	<a href="#">VariationPointProxy</a>	*	aggr	Proxy of a variation points in the C/C++ implementation. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=variationPointProxy.shortName

**Table 10.4: BswInternalBehavior**



# 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 <i>n</i> software-components, e.g. used for ECU mode.
A*	1..n	RPort:PPort	Currently not used, not supported for client-server communication.
A**	1..n	PRPort:RPort	Distribution of data or modes to <i>n</i> software-components, e.g. used for ECU mode.
B	1..1	PPort:RPort	Software-component 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 PortDefinedArgumentValues.
C	n..1	RPort:PPort	Software-component acts as Server, used for so called “call-backs” invoked by more than one Service.
D	1..1	PR-Port:PRPort	I/O control data.

**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 [10]). 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` [10], 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, C or D 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`<sup>1</sup>.]()

- **[TPS\_SWCT\_01409] Update of `PortDefinedArgumentValues`** [Depending on the configuration of the `Service` BSW it might be necessary to update the `ValueSpecifications` belonging to the `PortDefinedArgumentValues` generated in a previous step.]()

---

<sup>1</sup>This 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`.

<b>Class</b>	<b>SwcBswMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::SwcBswMapping			
<b>Note</b>	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. <b>Tags:</b> atp.recommendedPackage=SwcBswMappings			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
bswBehavior	<a href="#">BswInternalBehavior</a>	0..1	ref	The mapped BswInternalBehavior
runnable Mapping	SwcBswRunnable Mapping	*	aggr	A mapping between a pair of SWC and BSW runnables. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
swcBehavior	<a href="#">SwcInternalBehavior</a>	0..1	ref	The mapped SwcInternalBehavior.
synchronized ModeGroup	SwcBswSynchronized ModeGroupPrototype	*	aggr	A pair of SWC and BSW mode group prototypes to be synchronized by the scheduler. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
synchronized Trigger	SwcBswSynchronized Trigger	*	aggr	A pair of SWC and BSW Triggers to be synchronized by the scheduler. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 11.2: SwcBswMapping**

## 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 `<<atpSplitable>>` 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.

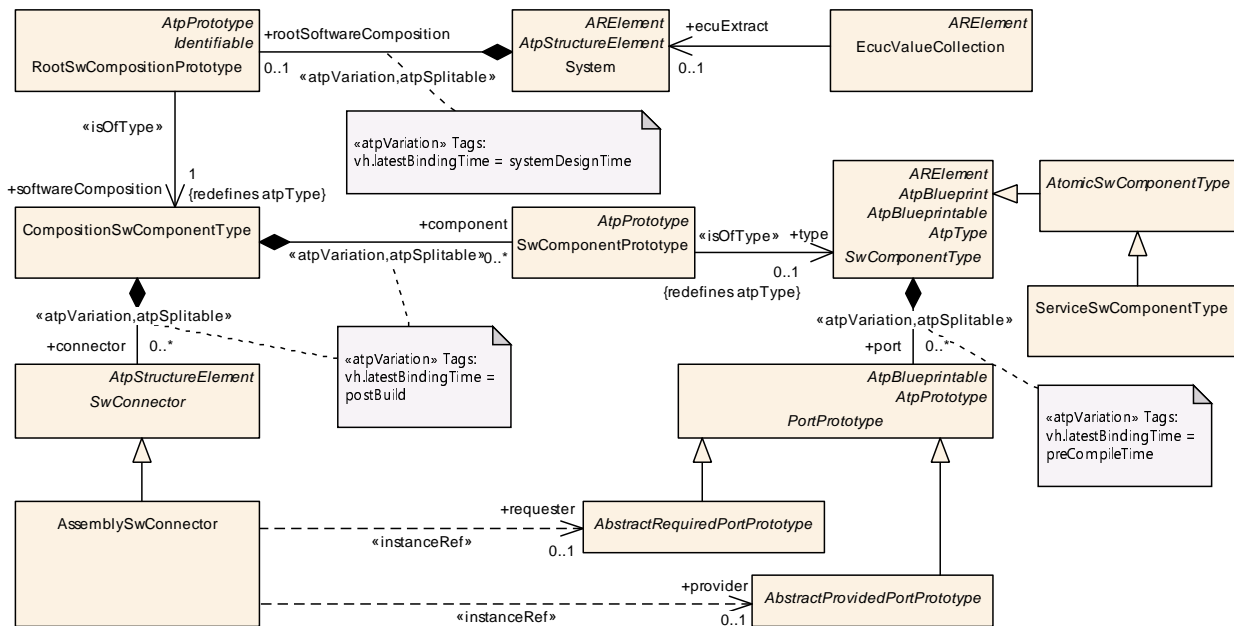


Figure 11.1: Usage of **RootSwCompositionPrototype** on an ECU

### 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` **at the time when the RTE is generated.**

The exceptions described in

- [TPS\_SWCT\_01572],
- [TPS\_SWCT\_01579],
- [TPS\_SWCT\_01831] and
- [TPS\_SWCT\_01580]

apply.]()

**[TPS\_SWCT\_01579] Dcm can directly access SenderReceiverInterface.dataElements, NvDataInterface.nvDataS, or ParameterInterface.parameters in AbstractProvidedPortPrototype** [An exception to the rule described in [constr\_2019] applies: the `Dcm` can directly access *SenderReceiverInterface*.

`dataElements`, `NvDataInterface.nvDatas`, or `ParameterInterface.parameters` in `AbstractProvidedPortPrototypes` (`PPortPrototype` in case of `ParameterInterface.parameter`) as long as `[constr_1071]` is fulfilled.

For this purpose, the `ServiceSwComponentType` that represents the `Dcm` functionality can have `AbstractRequiredPortPrototypes` typed by a compatible `SenderReceiverInterface` that may set attribute `isService` to `FALSE`.`()`

Please note that the inclusion of `ParameterInterface.parameters` in `[TPS_SWCT_01579]` is done under the assumption that the `ServiceSwComponentType` that represents the `Dcm` would connect to a `PPortPrototype` owned by a `ParameterSwComponentType`.

**`[TPS_SWCT_01831]` `Dcm` can directly access `SenderReceiverInterface.dataElements` or `NvDataInterface.nvDatas` in `AbstractRequiredPortPrototypes`** [An exception to the rule described in `[constr_2019]` applies: the `Dcm` can directly access `SenderReceiverInterface.dataElements` or `NvDataInterface.nvDatas` in `AbstractRequiredPortPrototypes` as long as `[constr_1071]` is fulfilled.

For this purpose, the `ServiceSwComponentType` that represents the `Dcm` functionality can have `AbstractProvidedPortPrototypes` typed by a compatible `SenderReceiverInterface` that may set attribute `isService` to `FALSE`.`()`

Please note that the exclusion of `ParameterInterface.parameters` from `[TPS_SWCT_01831]` is a direct consequence of `[constr_1137]`, i.e. the `ServiceSwComponentType` that represents the `Dcm` is not allowed to expose a `PPortPrototype` typed by a `ParameterInterface`.

**`[TPS_SWCT_01580]` `Dem` can directly access `SenderReceiverInterface.dataElements`, `NvDataInterface.nvDatas`, or `ParameterInterface.parameters` in `PPortPrototypes`** [An exception to the rule described in `[constr_2019]` applies: the `Dem` can directly access `SenderReceiverInterface.dataElements`, `NvDataInterface.nvDatas`, or `ParameterInterface.parameters` in `AbstractProvidedPortPrototypes` (`PPortPrototype` in case of `ParameterInterface.parameter`) as long as `[constr_1071]` is fulfilled.

For this purpose, the `ServiceSwComponentType` that represents the `Dem` functionality can have `RPortPrototypes` typed by a compatible `SenderReceiverInterface` that may set attribute `isService` to `FALSE`.`()`

Please note that the inclusion of `ParameterInterface.parameters` in `[TPS_SWCT_01580]` is done under the assumption that the `ServiceSwComponentType` that represents the `Dem` would connect to a `PPortPrototype` owned by a `ParameterSwComponentType`.

**`[TPS_SWCT_01411]` Use cases for a `ServiceSwComponentType` to express `ServiceNeeds`** [There are valid use cases for a `ServiceSwComponentType` to

express *ServiceNeeds*<sup>2</sup>. 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 [6].

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 [6].

**[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 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*.]( )

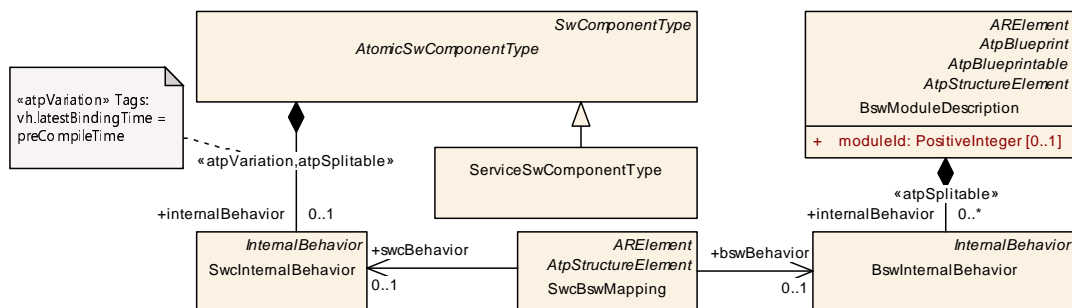


Figure 11.2: *ServiceSwComponentType*

<sup>2</sup>Thereby the previously existing constraint 1127 becomes invalid.

<b>Class</b>	<b>ServiceSwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	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. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtomicSwComponentType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 11.3: ServiceSwComponentType**

In the process of creating [PortPrototypes](#) the specified communication pattern A, B, or C for a specific kind of service port has to be considered, see table 11.1.

**[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 [21]) is part of the basic software but conceptually still needs to communicate with application software components located on other ECUs.

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](#).]()

The concept of mode requests across ECU boundaries is exemplified in Figure 11.3.

**[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](#).]()



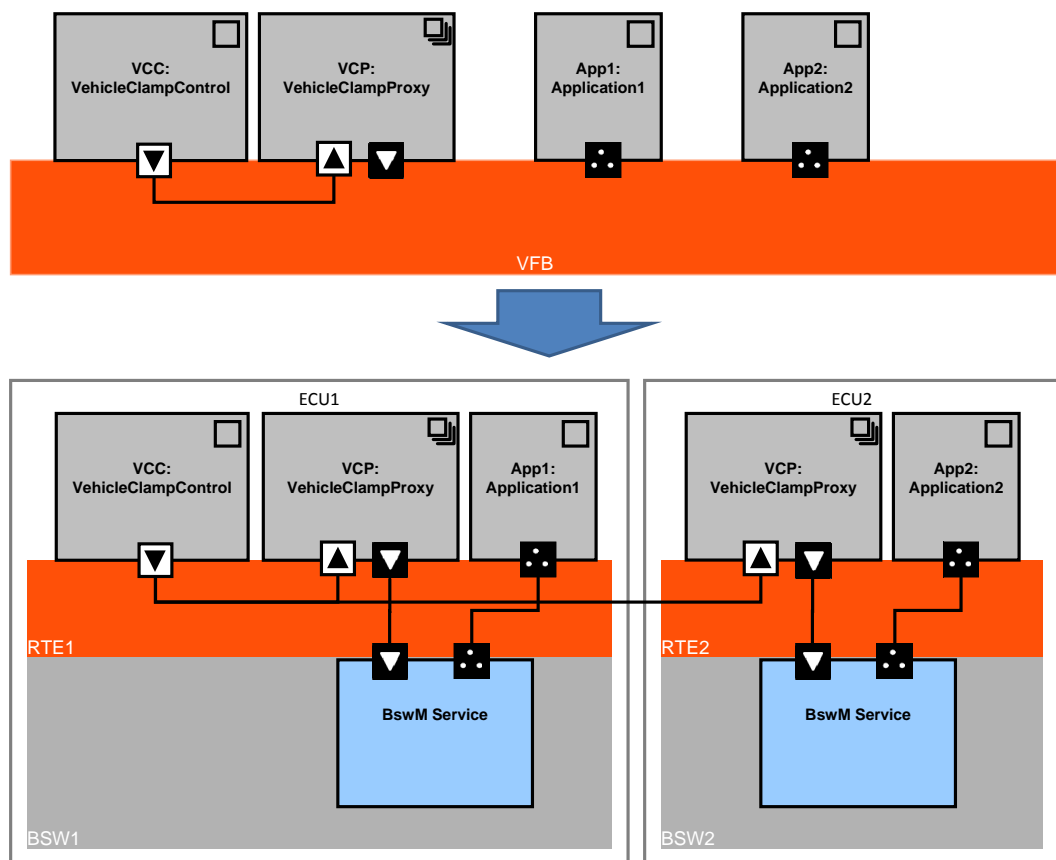


Figure 11.3: Mode request over the network [3]

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. ]()

**[constr\_2016] Connections between `SwComponentPrototypes` of type `ServiceProxySwComponentType`** [A connection between `PortPrototypes` belonging to `SwComponentPrototypes` where both are typed by `ServiceProxySwComponentType` is not permitted **at the time when the RTE is generated.** ]()

<b>Class</b>	<b>ServiceProxySwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	<p>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.</p> <p>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.</p> <p>Apart from the semantics, a ServiceProxySwComponentType has these specific properties:</p> <ul style="list-style-type: none"> <li>• A prototype of it can be mapped to more than one ECUs in the system description.</li> <li>• Exactly one additional instance of it will be created in the ECU-Extract per ECU to which the prototype has been mapped.</li> <li>• For remote communication, it can have only R-Ports with sender-receiver interfaces and 1:n semantics.</li> <li>• There shall be no connectors between two prototypes of any ServiceProxySwComponentType.</li> </ul> <p><b>Tags:</b>atp.recommendedPackage=SwComponentTypes</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtomicSwComponentType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 11.4: ServiceProxySwComponentType**

**[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.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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.

This rule shall be imposed **at the time when the RTE is generated.**]()

## 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` [35]. 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 13.

- 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` [35] can be found in section 13.2.

### 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*)

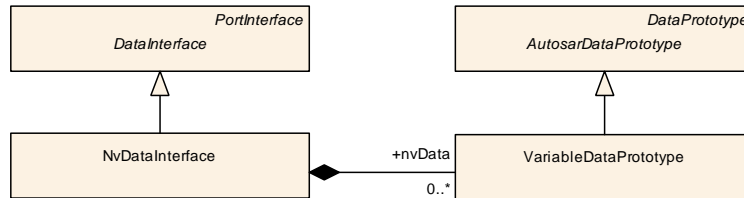
**[TPS\_SWCT\_01143] Non-volatile data represented by an `NvBlockSwComponentType` can be read and written** [The non-volatile data represented by an `NvBlockSwComponentType` can be read and written. For this purpose the `NvBlockSwComponentType` is allowed to have `PPortPrototypes` and `RPortPrototypes`.] (*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`

This rule shall be imposed **at the time when the RTE is generated.**]()



**Figure 11.4: NvDataInterface**

**[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 **at the time when the RTE is generated.**]()

<b>Class</b>	<b>NvBlockSwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	The NvBlockSwComponentType defines non volatile data which data can be shared between Sw ComponentPrototypes. The non volatile data of the NvBlockSwComponentType are accessible via provided and required ports. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<i>ARElement, ARObject, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
bulkNvData Descriptor	<a href="#">BulkNvDataDescriptor</a>	*	aggr	This aggregation formally defines the bulk Nv Blocks that are provided to the application software by the enclosing NvBlockSwComponentType. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=bulkNvDataDescriptor.shortName, bulkNvDataDescriptor.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
nvBlock Descriptor	<a href="#">NvBlockDescriptor</a>	*	aggr	Specification of the properties of exactly one NVRAM Block. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=nvBlockDescriptor.shortName, nvBlockDescriptor.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

**Table 11.5: NvBlockSwComponentType**

<b>Class</b>	<b>NvDataInterface</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface





<b>Class</b>	<b>NvDataInterface</b>			
<b>Note</b>	A non volatile data interface declares a number of VariableDataPrototypes to be exchanged between non volatile block components and atomic software components. <b>Tags:</b> atp.recommendedPackage=PortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DataInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
nvData	<a href="#">VariableDataPrototype</a>	*	aggr	The VariableDataPrototype of this nv data interface.

**Table 11.6: NvDataInterface**

### 11.5.3 Software-Components using *NVRAM data* of NvBlockComponents

[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 value<sup>3</sup>.] ([RS\\_SWCT\\_03225](#))

[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` **at the time when the RTE is generated**.] ()

[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](#) [28]. [constr\_1148] applies.

This rule shall be imposed **at the time when the RTE is generated**.] ()

For details see chapter 6.4.4.

<sup>3</sup>Note 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 [ApplicationSwComponentType](#) and the [SensorActuatorSwComponentType](#) will be candidates for using this feature but it will obviously not be reasonable for e.g. [NvBlockSwComponentType](#).

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.

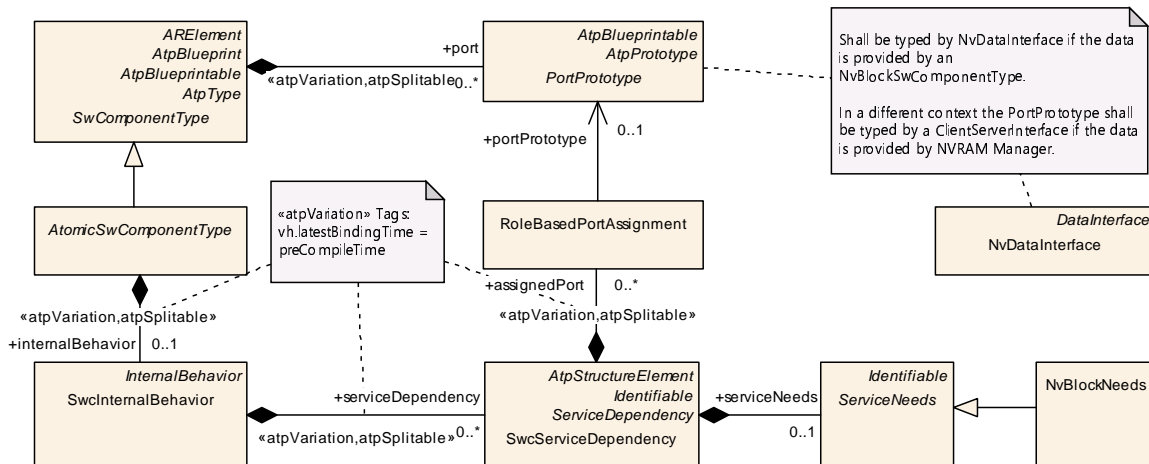


Figure 11.5: NvBlockNeeds for *nv data* PortPrototypes

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 13.2.4.

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 NVRAM Block 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.

**[TPS\_SWCT\_01675] Recommendations for attributes of NvBlockNeeds or for NvBlockDescriptor** [The formal modeling of a `NvBlockDescriptor` should follow the recommendations given in table 11.7.](RS\_SWCT\_03225)

But please note that table 11.7 does not represent a binding constraint.

Attribute of <code>NvBlockNeeds</code>	<code>NvBlockNeeds</code> of different <code>nv data Port-Prototypes</code> of software-components	<code>NvBlockNeeds</code> of <code>NvBlockDescriptor</code>
<code>readonly</code>	Recommended to match for all connected <code>nv data PortPrototypes</code> if specified.	Recommended to be identical as requested by <code>nv data PortPrototypes</code> .
<code>reliability</code>	Can be different.	Recommended to be set to the highest reliability class request by any mapped <code>nv data PortPrototypes</code> .
<code>resistantToChangedSw</code>	Recommended to match for all connected <code>nv data PortPrototypes</code> if specified.	Recommended to be identical as requested by <code>nv data PortPrototypes</code> .
<code>restoreAtStart</code>	Recommended to match for all connected <code>nv data PortPrototypes</code> if specified.	Recommended to be identical as requested by <code>nv data PortPrototypes</code> .
<code>storeAtShutdown</code>	Recommended to match for all connected <code>nv data PortPrototypes</code> if specified.	Recommended to be identical as requested by <code>nv data PortPrototypes</code> .
<code>writeOnlyOnce</code>	Recommended to match for all connected <code>nv data PortPrototypes</code> if specified.	Recommended to be identical as requested by <code>nv data PortPrototypes</code> .
<code>writingFrequency</code>	Can be different.	Recommended to be set to the highest requested frequency of the mapped <code>nv data PortPrototypes</code> .
<code>writingPriority</code>	Can be different.	Recommended to be set to the highest requested priority of the mapped <code>nv data Port-Prototypes</code> .
<code>writeVerification</code>	Can be different.	Recommended to set to true if any of the <code>nv data PortPrototypes</code> requests a write verification.
<code>calcRamBlockCrc</code>	Can be different.	Recommended to set to true if any of the <code>nv data PortPrototypes</code> requests a CRC calculation.
<code>checkStaticBlockId</code>	Can be different.	Recommended to set to true if any of the <code>nv data PortPrototypes</code> requests a check of the static block ID.
<code>ramBlockStatusControl</code>	Can be different.	Recommended to set to <code>RamBlockStatusControlEnum.api</code> if any of the <code>nv data PortPrototypes</code> requests a use of the API for accessing the block.
<code>storeCyclic</code>	Can be different.	Recommended to set to true if any of the <code>nv data PortPrototypes</code> requests cyclic writing.
<code>storeEmergency</code>	Can be different.	Recommended to set to true if any of the <code>nv data PortPrototypes</code> requests emergency writing.
<code>storeImmediate</code>	Can be different.	Recommended to set to true if any of the <code>nv data PortPrototypes</code> requests immediate writing.
<code>storeOnChange</code>	Can be different.	Recommended to set to true if any of the <code>nv data PortPrototypes</code> requests on-change writing.
<code>selectBlockForFirstInitAll</code>	Recommended to match for all connected <code>nv data PortPrototypes</code> if specified.	Recommended to be identical as requested by <code>nv data PortPrototypes</code> .

**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 Software-Components connected to NvBlockComponents

Please note that restrictions apply on the creation of `AssemblySwConnectors` between `NvBlockSwComponentType` and other `AtomicSwComponentTypes`.

In particular `ApplicationSwComponentTypes` communicating with each other used buffers generated and controlled by the RTE to exchange data. An `NvBlockSwComponentType`, however, maintains its own buffer in form of the `ramBlock`.

Thus, an `ApplicationSwComponentType` that reads a `dataElement` that may be provided by either another `ApplicationSwComponentType` or an `NvBlockSwComponentType` could not actually access the `dataElement` because it cannot decide whether it needs to access the buffer provided by the RTE or the `ramBlock`.

Therefore, scenarios like this are considered invalid by regulation of the AUTOSAR standard.

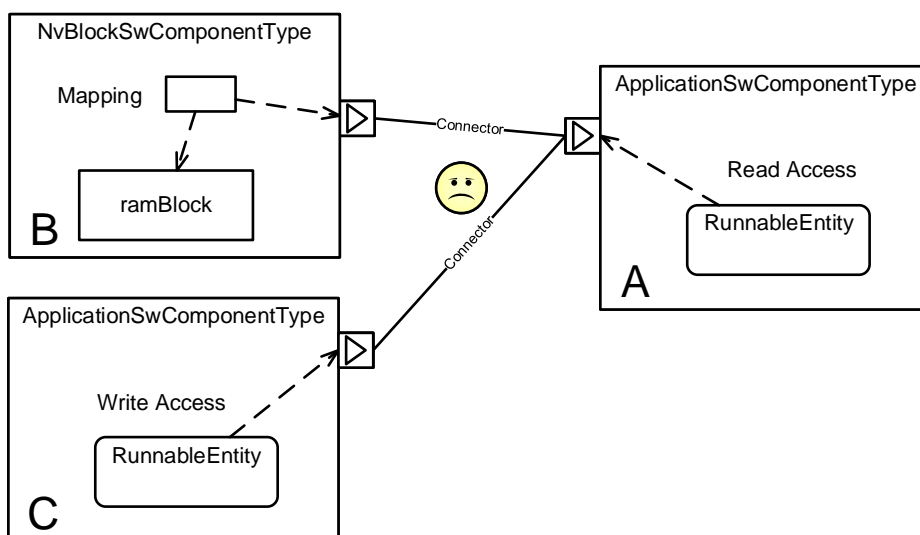


Figure 11.6: Example invalid connection between software-components (a)

**[constr\_1417] Invalid connection between `NvBlockSwComponentType` and other `AtomicSwComponentType` (I)** [A configuration where an `RPortPrototype` owned



by an `AtomicSwComponentType` is simultaneously and directly connected to `AbstractProvidedPortPrototypes` of a collection of `AtomicSwComponentTypes` where at least one in the collection is an `NvBlockSwComponentType` for a matching set of `dataElements` in all these `PortPrototypes` shall be considered invalid **at the time when the RTE is generated.** ]()

The scenario covered by [constr\_1417] is depicted in Figures 11.6 and 11.7.

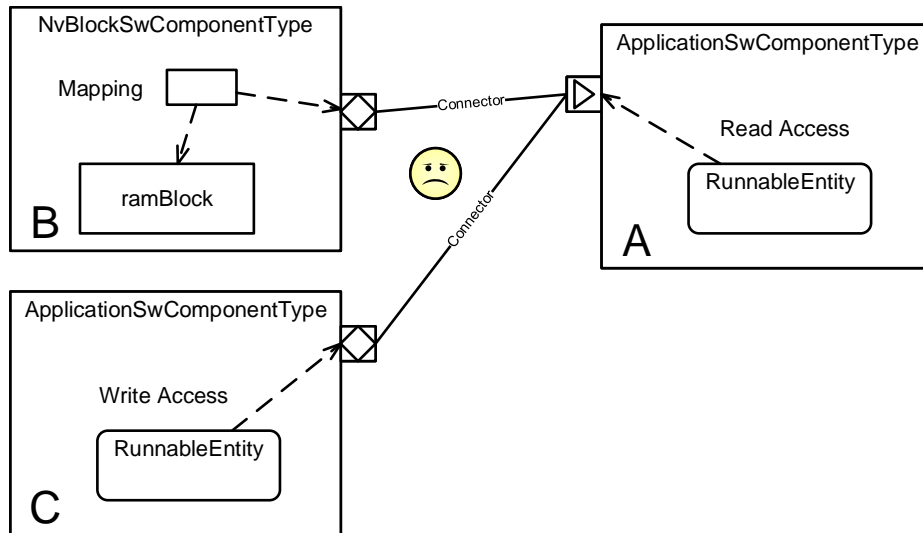


Figure 11.7: Example invalid connection between software-components (b)

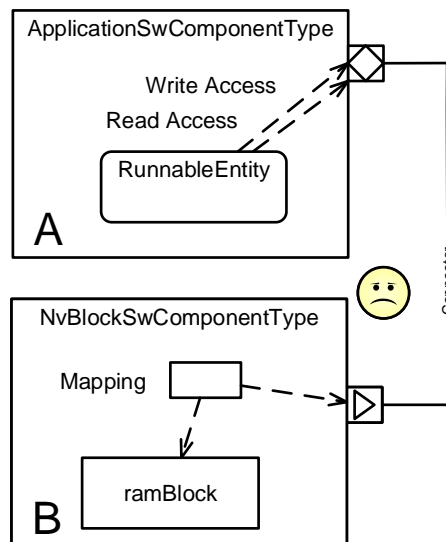


Figure 11.8: Example invalid connection between software-components (c)

**[constr\_1418] Invalid connection between `NvBlockSwComponentType` and other `AtomicSwComponentType` (II)** [A configuration where a `PRPortPrototype` owned by an `AtomicSwComponentType` is connected to a `PPortPrototype` owned by an

`NvBlockSwComponentType` for a matching set of `dataElements` in all these `PortPrototypes` shall be considered invalid **at the time when the RTE is generated.**]()

The scenario covered by [constr\_1418] is depicted in Figure 11.8.

### 11.5.5 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`.]()

<b>Class</b>	<b>NvBlockDescriptor</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::NvBlockComponent			
<b>Note</b>	Specifies the properties of exactly on <code>NVRAM Block</code> .			
<b>Base</b>	<code>ARObject</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
clientServerPort	<code>RoleBasedPortAssignment</code>	*	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.  <b>Stereotypes:</b> <code>atpVariation</code> <b>Tags:</b> <code>vh.latestBindingTime=preCompileTime</code>
constantValueMapping	<code>ConstantSpecificationMappingSet</code>	*	ref	Reference to the <code>ConstantSpecificationMapping</code> to be applied for the particular <code>NVRAM Block</code>  <b>Stereotypes:</b> <code>atpSplitable</code> <b>Tags:</b> <code>atp.Splitkey=constantValueMapping</code>
dataTypeMapping	<code>DataTypeMappingSet</code>	*	ref	Reference to the <code>DataTypeMapping</code> to be applied for the particular <code>NVRAM Block</code> .  <b>Stereotypes:</b> <code>atpSplitable</code> <b>Tags:</b> <code>atp.Splitkey=dataTypeMapping</code>





Class	NvBlockDescriptor			
instantiationDataDefProps	<a href="#">InstantiationDataDefProps</a>	*	aggr	<p>The purpose of InstantiationDataDefProps are the refinement of some data def properties of individual instantiations within the context of a NvBlockSw ComponentType.</p> <p>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.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
modeSwitchEventTriggeredActivity	<a href="#">ModeSwitchEventTriggeredActivity</a>	*	aggr	<p>This represents the collection of ModeSwitchEventTriggeredActivities related to the enclosing NvBlockDescriptor.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=modeSwitchEventTriggeredActivity, modeSwitchEventTriggeredActivity.variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
nvBlockDataMapping	<a href="#">NvBlockDataMapping</a>	*	aggr	<p>Defines the mapping between the VariableDataPrototypes in the NvBlockComponents ports and the VariableDataPrototypes of the RAM Block.</p> <p>The aggregation of NvBlockDataMapping is subject to variability with the purpose to support the conditional existence of nv data ports.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
nvBlockNeeds	<a href="#">NvBlockNeeds</a>	0..1	aggr	<p>Specifies the abstract needs on the configuration of the NVRAM Manager for the single NVRAM Block described by this NvBlockDescriptor.</p> <p>In addition, it may define requirements for writing strategies in an implementation of an NvBlockSw ComponentType by the RTE.</p> <p>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.</p>
ramBlock	<a href="#">VariableDataPrototype</a>	0..1	aggr	<p>Defines the RAM Block of the NVRAM Block provided by NvBlockSwComponentType.</p>
romBlock	<a href="#">ParameterDataPrototype</a>	0..1	aggr	<p>Defines the ROM Block of the NVRAM Block provided by NvBlockSwComponentType.</p>
supportDirtyFlag	Boolean	0..1	attr	<p>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.</p>
timingEvent	<a href="#">TimingEvent</a>	0..1	ref	<p>this reference can be taken to identify the TimingEvent to be used by the RTE for implementing a cyclic writing strategy for this block</p>
writingStrategy	<a href="#">RoleBasedDataAssignment</a>	*	aggr	<p>This attribute allows for assigning a specific writing strategy for an incoming AutosarDataPrototype.</p>

**Table 11.8: NvBlockDescriptor**

**[constr\_1981] Existence of attribute [NvBlockDescriptor.nvBlockNeeds](#)** [For each [NvBlockDescriptor](#), attribute [nvBlockNeeds](#) shall exist **at the time when the RTE is generated.** ]()

For more explanation about the semantics of the attribute `NvBlockDescriptor.supportDirtyFlag` please refer to the SWS RTE [2].

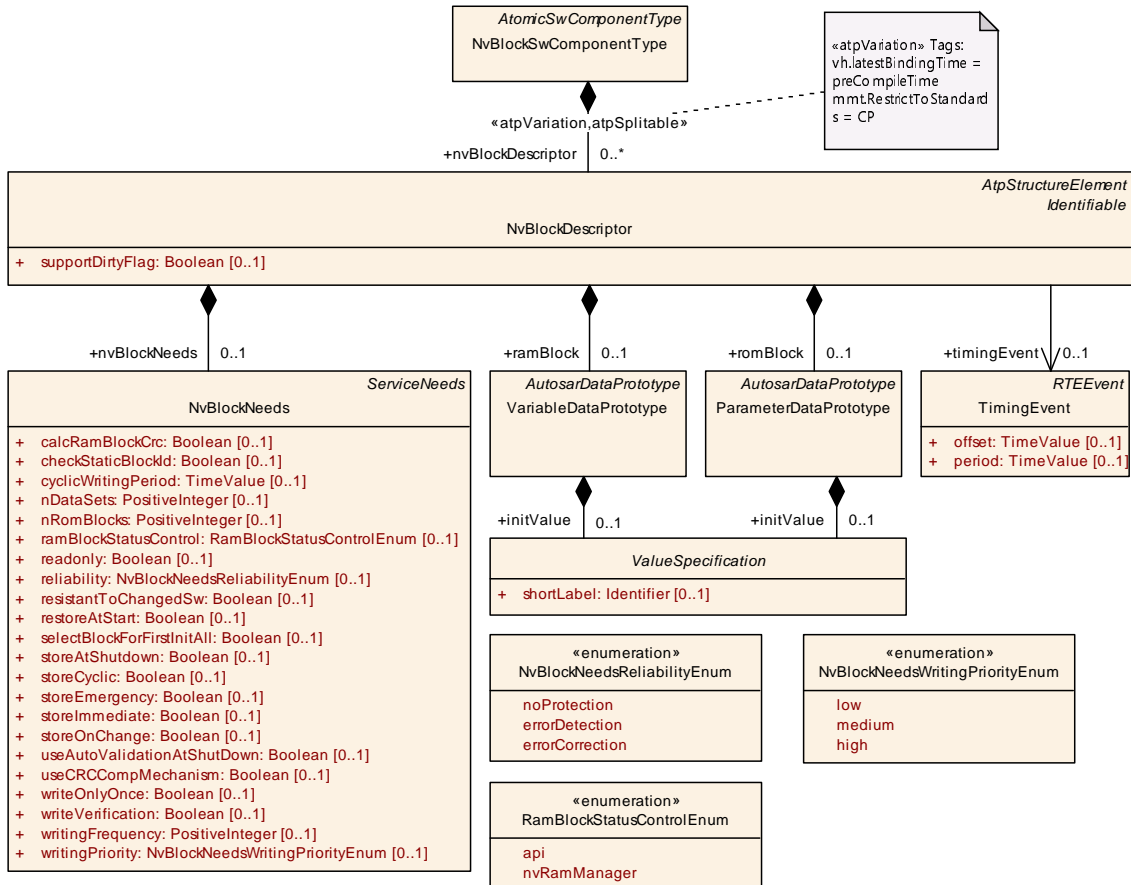


Figure 11.9: `NvBlockSwComponentType` and `NvBlockDescriptor`

**[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` at the time when the RTE is generated.]()

The reason for the existence of [constr\_1095] is that the AUTOSAR NvM [35] 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`.

### 11.5.5.1 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.nvBlockNeeds.cyclicWritingPeriod` needs to exist and have a reasonable value.

In the context of using the attribute `NvBlockDescriptor.nvBlockNeeds.cyclicWritingPeriod` the constraints [constr\_1308] and [constr\_1309] apply.

- 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`.

- 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 is written to NVRAM storage when the ECU shuts down.

This use case corresponds to setting the value of attribute `NvBlockDescriptor.nvBlockNeeds.storeAtShutdown` to `true`.

- Write on **mode switch**. Here, the data is written to NVRAM in response to a mode switch configured to trigger the writing.

This use case corresponds to the existence of attribute `NvBlockDescriptor.modeSwitchEventTriggeredActivity`.

- Write on **change**. Here, the data is written to NVRAM if the value is different than the respective value in the `ramMirror`.

This use case corresponds to setting the value of attribute `NvBlockDescriptor.nvBlockNeeds.storeOnChange` to `true`.

](RS\_SWCT\_03225)

Please refer to [TPS\_SWCT\_01587] and Figure 11.10 for more information about how the use case to write data cyclically can be configured.

Please refer to [TPS\_SWCT\_01588] and Figure 11.11 for more information about how the use case to write data immediately can be configured.

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 11.10.

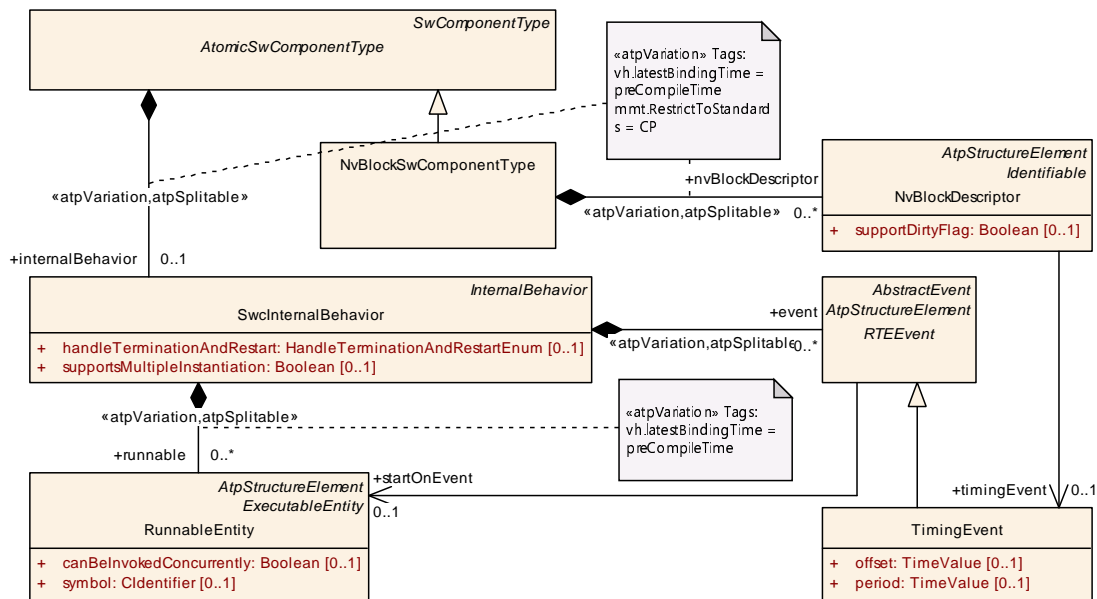


Figure 11.10: How to model a cyclic writing strategy for *nv data*

[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 11.11.

[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** [35], e.g. in [SWS\_NvM\_00182] and [SWS\_NvM\_00300].

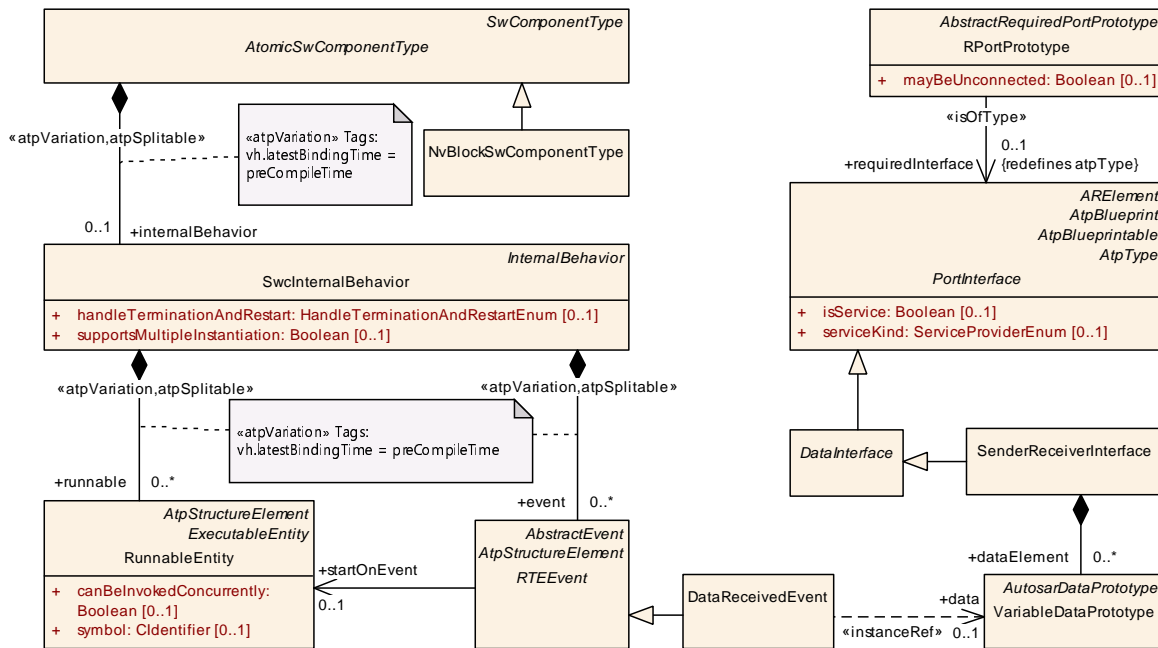


Figure 11.11: How to model an immediate writing strategy for *nv data*

[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.nvBlockNeeds.storeImmediate` and `NvBlockDescriptor.nvBlockNeeds.storeCyclic` may exist and set to `true` in the context of the same `NvBlockNeeds`.

[TPS\_SWCT\_01665] **Usage of `SwcModeSwitchEvent` for triggering a write procedure of *nv data*** [The approach to manage data of an `NvBlockSwComponentType` in response to a mode switch notification received from a mode manager requires the activation of a `RunnableEntity` by a `SwcModeSwitchEvent`.] (RS\_SWCT\_03225)

[TPS\_SWCT\_01666] **Semantics of `ModeSwitchEventTriggeredActivity.role`** [If the role `ModeSwitchEventTriggeredActivity.role` is set to the value `WriteBlock` then NvM gets requested to write the *nv data* block after the corresponding `SwcModeSwitchEvents` occurs.] (RS\_SWCT\_03225)

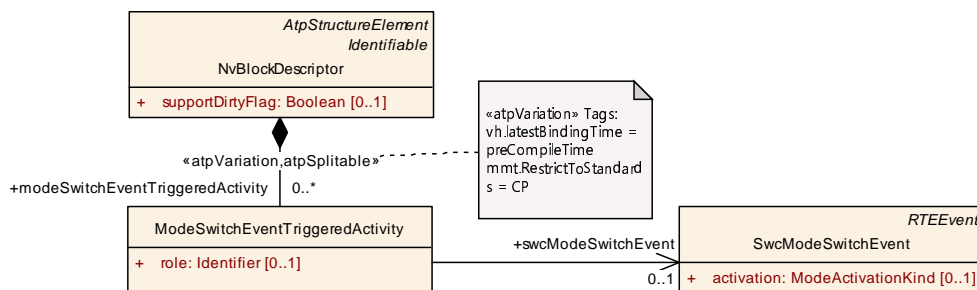


Figure 11.12: Usage of mode switch notification for the activation of a write-procedure of *nv data*

**[constr\_1415] Supported values of ModeSwitchEventTriggeredActivity.role** [The only supported value of ModeSwitchEventTriggeredActivity.role at the time when the RTE is generated is WriteBlock.]()

<b>Class</b>	<b>ModeSwitchEventTriggeredActivity</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::NvBlockComponent			
<b>Note</b>	This meta-class defines an activity of the NvBlockSwComponentType for a specific NvBlock which is triggered by a ModeSwitchEvent.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
role	Identifier	0..1	attr	This attribute indicates which service of the NvM for the NvBlock shall be requested.
swcModeSwitchEvent	SwcModeSwitchEvent	0..1	ref	This reference identifies the SwcModeSwitchEvent that triggers the activity.

**Table 11.9: ModeSwitchEventTriggeredActivity**

**[constr\_1982] Existence of attribute ModeSwitchEventTriggeredActivity.role** [For each ModeSwitchEventTriggeredActivity, attribute role shall exist at the time when the RTE is generated.]()

**[constr\_1983] Existence of attribute ModeSwitchEventTriggeredActivity.swcModeSwitchEvent** [For each ModeSwitchEventTriggeredActivity, attribute swcModeSwitchEvent shall exist at the time when the RTE is generated.]()

### 11.5.5.1.1 Existence of multiple event-driven Writing Strategies

A use case might exist where (potentially in addition to a cyclic writing strategy) multiple event-based writing strategies to NvRam exist at the same time.

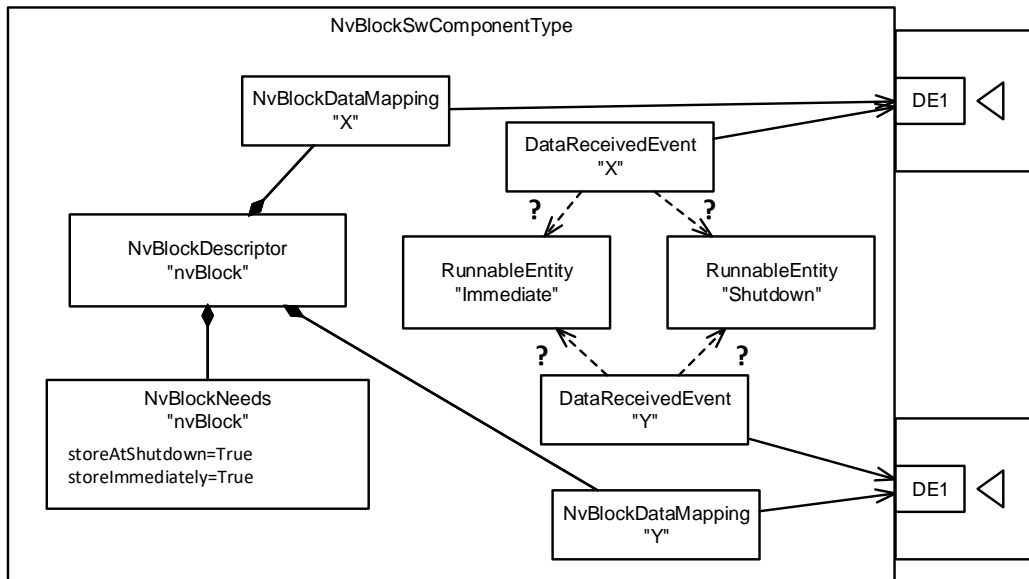
Writing strategies are typically implemented by means of different RunnableEntities that need to be triggered by the arrival of data at the NvBlockSwComponentType. In this case, however, it is necessary to be able to assign the relation between incoming data and the RunnableEntity that applies the fitting writing strategy.

In the example scenario depicted in Figure 11.13, two dataElements are received by the NvBlockSwComponentType. One dataElement shall be associated with the writing strategy to store at shutdown and the other dataElement shall be associated with an immediate storage.

Two DataReceivedEvents exist that can be taken to trigger RunnableEntities. But without further formal specification, it is up to speculation which DataReceivedEvent shall be associated with which RunnableEntity.

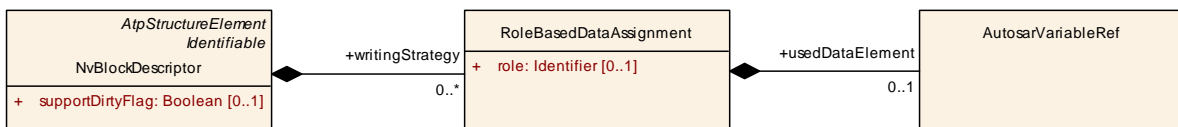
To remove this degree of uncertainty, it is possible to use standardized values for the attribute RoleBasedDataAssignment.role and let the RoleBasedDataAssignment itself refer to the respective dataElement.





**Figure 11.13: Existence of several event-based writing strategies for the same `NvBlockDescriptor`**

The approach is sketched in Figure 11.14 and its application to the depicted scenario is sketched in Figure 11.15.



**Figure 11.14: Modeling of `RoleBasedDataAssignment` in the context of `NvBlockDescriptor`**

[TPS\_SWCT\_01795] Further specification to facilitate the association of writing strategies to the corresponding `RunnableEntity` [`NvBlockDescriptor` may aggregate `RoleBasedDataAssignment` in the role `writingStrategy` and the attribute `NvBlockDescriptor.writingStrategy.role` can be used to assign a writing strategy to an `AutosarDataPrototype` (referenced by means of `NvBlockDescriptor.writingStrategy.usedDataElement`) in the context of a `PortPrototype`.] (*RS\_SWCT\_03225*)

[constr\_1713] `NvBlockDescriptor.writingStrategy.usedDataElement` shall refer to `AutosarDataPrototype` [The reference `NvBlockDescriptor.writingStrategy.usedDataElement` shall **only** refer to an `AutosarDataPrototype` at the time when the RTE is generated.] ()

Rationale: the activation of a `RunnableEntity` that implements the respective writing strategy is triggered by the reception of an entire `AutosarDataPrototype`. The ability to refer to a `DataPrototype` inside the `AutosarDataPrototype` might give a false impression.

[constr\_1714] **AutosarDataPrototype** shall only be referenced by a single **NvBlockDescriptor.writingStrategy** [If an **AutosarDataPrototype** in the context of a **PortPrototype** is referenced from a **NvBlockDescriptor.writingStrategy** then this **AutosarDataPrototype** shall not be referenced from any other **NvBlockDescriptor.writingStrategy**.

This rule shall be imposed **at the time when the RTE is generated.** ]()

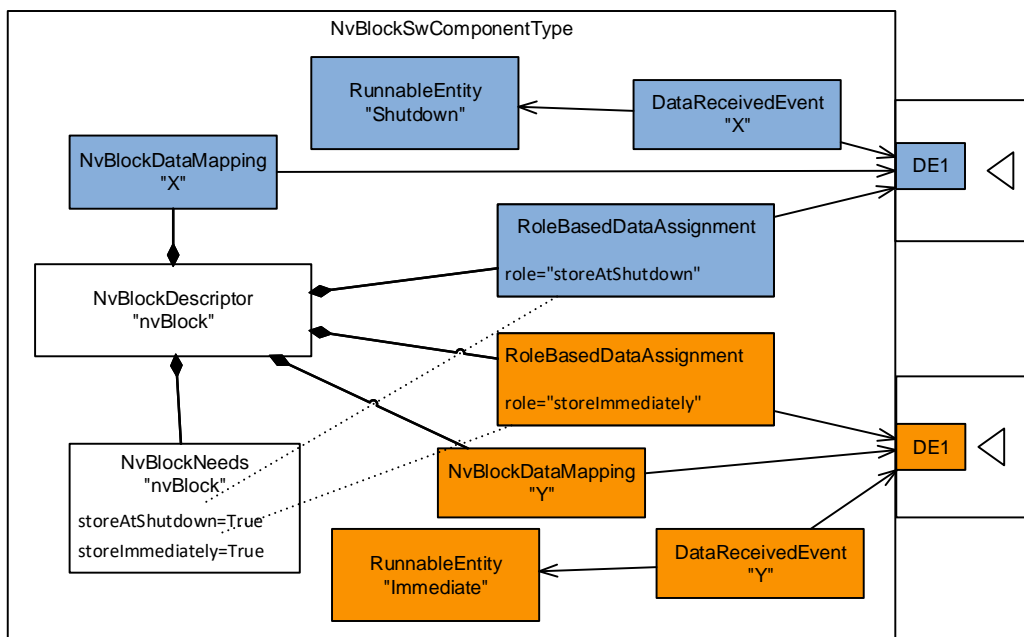


Figure 11.15: Solution for the existence of several event-based writing strategies for the same **NvBlockDescriptor**

[constr\_1715] Possible values of attribute **NvBlockDescriptor.writingStrategy.role** [The attribute **NvBlockDescriptor.writingStrategy.role** shall only have one of the following values **at the time when the RTE is generated** (see [TPS\_SWCT\_01586]):

- **storeAtShutdown**
- **storeImmediate**
- **storeOnChange**

]()

[constr\_1716] Consistency of attribute **NvBlockDescriptor.writingStrategy.role** set to **storeAtShutdown** [The existence of **NvBlockDescriptor.writingStrategy** where attribute **role** is set to **storeAtShutdown** is only supported if **NvBlockDescriptor.nvBlockNeeds.storeAtShutdown** exists and is set to **True** **at the time when the RTE is generated.** ]()

**[constr\_1717] Consistency of attribute `NvBlockDescriptor.writingStrategy.role` set to `storeImmediate`** [The existence of `NvBlockDescriptor.writingStrategy` where attribute `role` is set to `storeImmediate` is only supported if `NvBlockDescriptor.nvBlockNeeds.storeImmediate` exists and is set to `True` **at the time when the RTE is generated.**]()

**[constr\_10074] Consistency of attribute `NvBlockDescriptor.writingStrategy.role` set to `storeOnChange`** [The existence of `NvBlockDescriptor.writingStrategy` where attribute `role` is set to `storeOnChange` is only supported if `NvBlockDescriptor.nvBlockNeeds.storeOnChange` exists and is set to `True` **at the time when the RTE is generated.**]()

### 11.5.5.2 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 13.

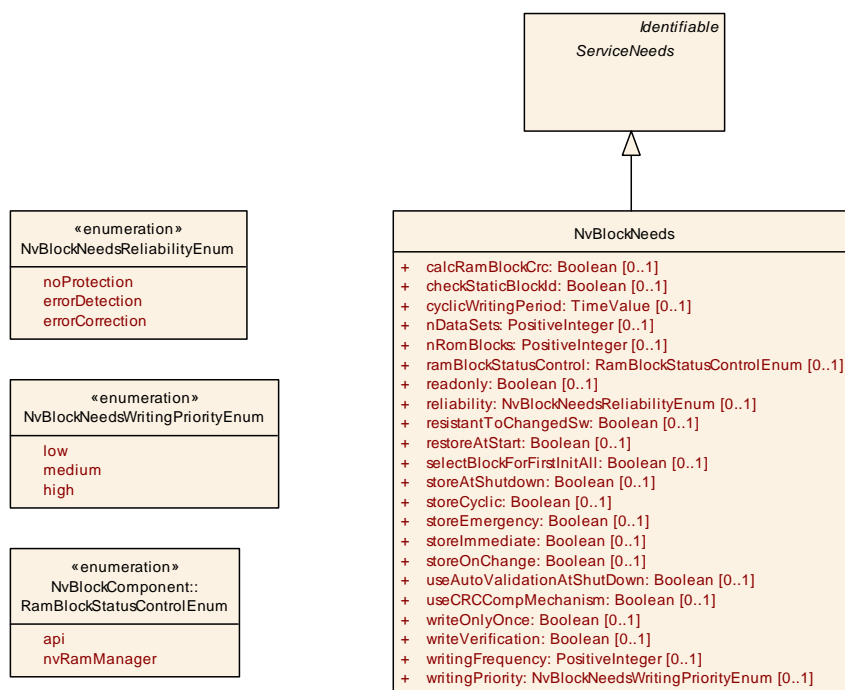


Figure 11.16: `NvBlockNeeds`

**[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`.

This rule shall be imposed **at the time when the RTE is generated**]()

<b>Class</b>	<b>NvBlockNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the abstract needs on the configuration of a single NVRAM Block.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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	<a href="#">TimeValue</a>	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	<a href="#">RamBlockStatusControlEnum</a>	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	<a href="#">NvBlockNeedsReliabilityEnum</a>	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.
selectBlockForFirstInitAll	Boolean	0..1	attr	If this attribute is set to true the NvM shall process this block in the NvM_FirstInitAll() function.
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.
storeOnChange	Boolean	0..1	attr	This attribute defines whether the associated RAM Block shall be stored immediately if the written value is different to the value stored in the associated RAM Block(s) during or after execution of the according SW-C RunnableEntity.
useAutoValidationAtShutdown	Boolean	0..1	attr	If set to true the RAM Block shall be auto validated during shutdown phase.
useCRCCompMechanism	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.





<i>Class</i>	<b>NvBlockNeeds</b>			
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 11.10: NvBlockNeeds**

<i>Enumeration</i>	<b>NvBlockNeedsReliabilityEnum</b>
<i>Package</i>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<i>Note</i>	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.
<i>Literal</i>	<i>Description</i>
errorCorrection	Errors shall be corrected <b>Tags:</b> atp.EnumerationLiteralIndex=0
errorDetection	Errors shall be detected <b>Tags:</b> atp.EnumerationLiteralIndex=1
noProtection	Data need not to be handled with protection <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 11.11: 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**
- **NvBlockNeeds.storeOnChange**

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**.

This rule shall be imposed **at the time when the RTE is generated.**]()

### 11.5.5.3 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** [35] 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 **at the time when the RTE is generated**, 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**.

### 11.5.5.4 NvBlockDataMapping

[TPS\_SWCT\_01148] **NvBlockDataMapping** [The meta-class **NvBlockDataMapping** specifies the mapping of (bits of) **DataPrototypes** of the **NvBlockSwComponentType**'s **PPortPrototypes** resp. **RPortPrototypes** to (bits of) **DataPrototypes** 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** [Unless both the attribute **bitfieldTextTableMaskNvBlockDescriptor** and attribute **bitfieldTextTableMaskPortPrototype** is defined in

the context of a given `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 `DataPrototype` aggregated by `NvBlockDataMapping` in the role `writtenNvData`, `writtenReadNvData`, or `readNvData`.

This rule shall be imposed **at the time when the RTE is generated.** ]()

**[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 **at the time when the RTE is generated**, 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 11.17.

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. `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 11.17).

**[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` **at the time when the RTE is generated.**]()

**[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 **at the time when the RTE is generated.**]()

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.



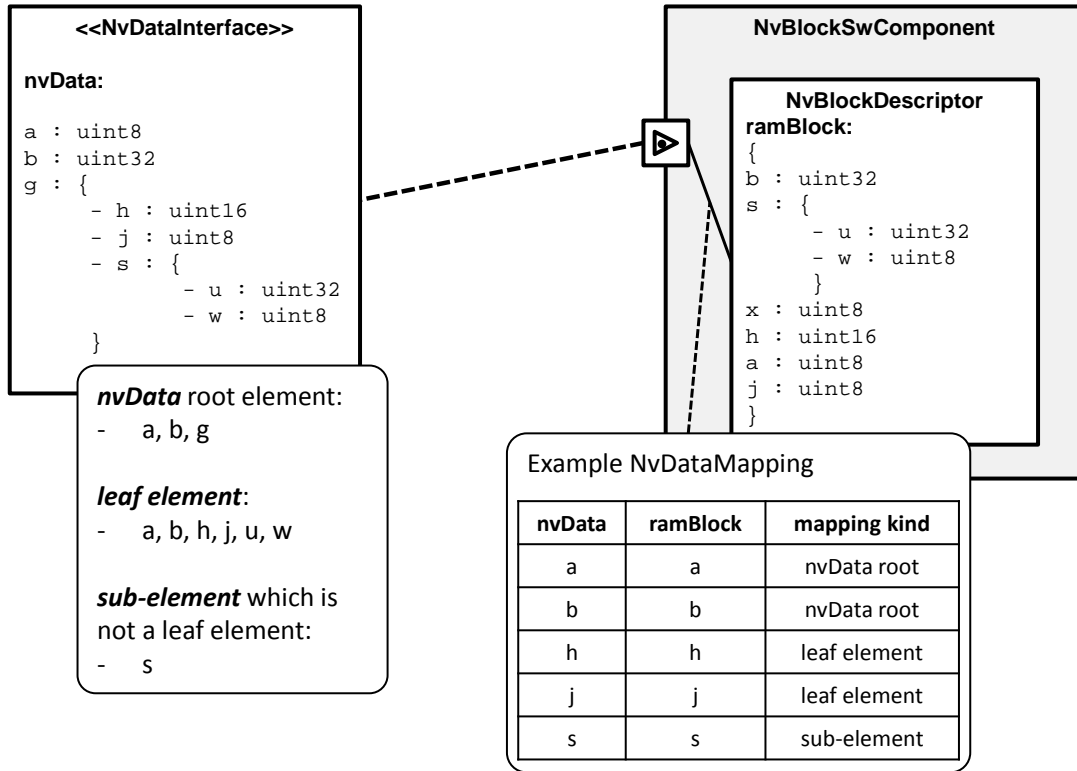


Figure 11.17: Example **NvBlockDataMapping** to explain the three ways to map a **VariableDataPrototype** to either an **NvBlockDescriptor.ramBlock** or a sub-element thereof

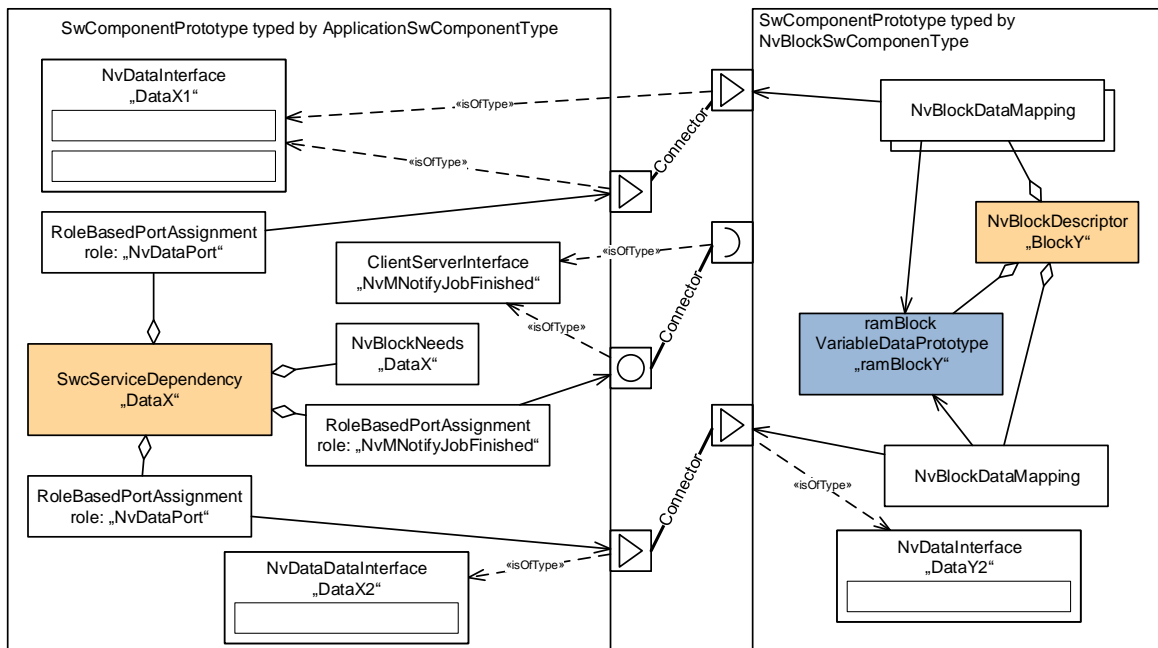


Figure 11.18: Visualization of the statement made by [constr\_1404]

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.

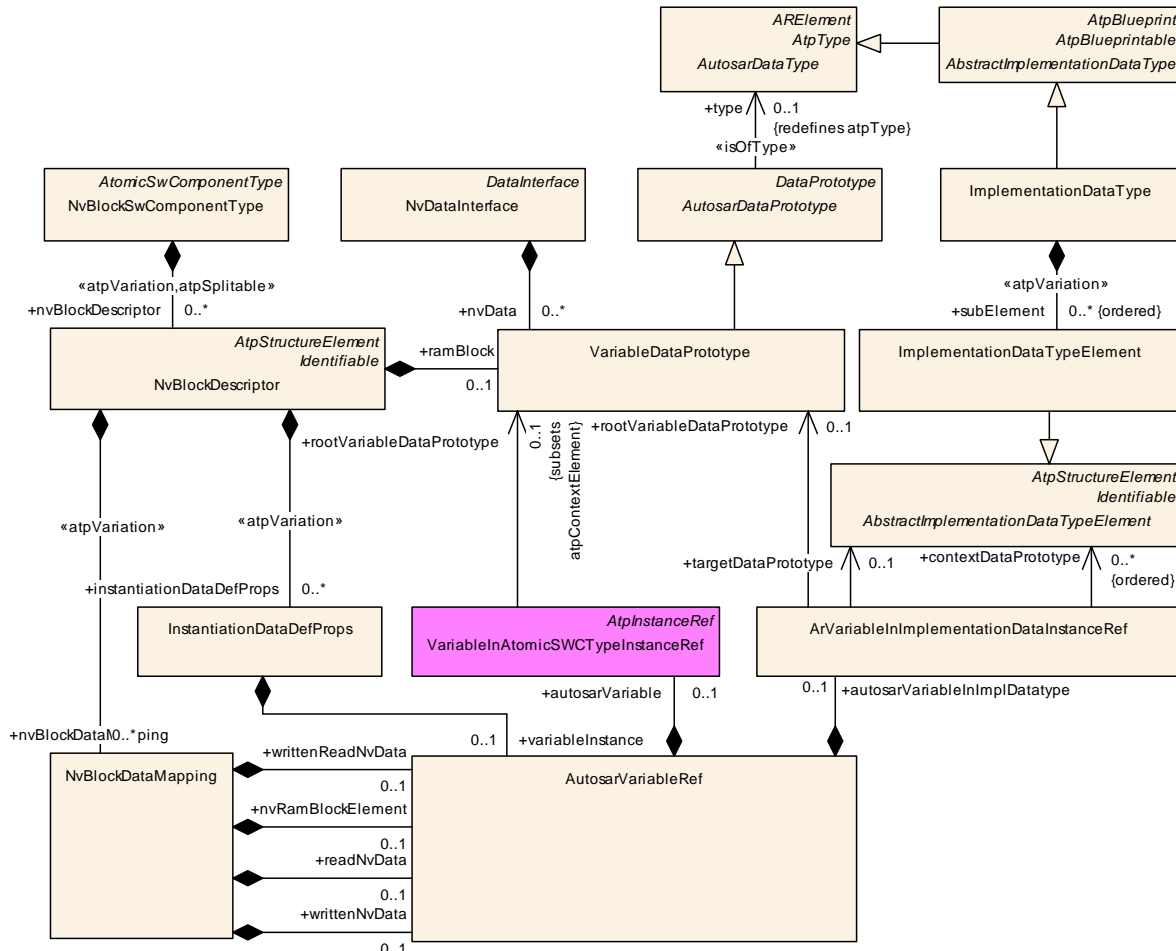


Figure 11.19: `NvBlockDataMapping` and `InstantiationDataDefProps`

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 aspect represents one motivation for 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 `NvBlockDescriptor` [In the context of a given `SwcServiceDependency` (which, in turn, is

owned by an `AtomicSwComponentType`), **all** `NvDataInterface.nvData` of `PortPrototypes` 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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

The statement made by [constr\_1404] is visualized in Figure 11.18. The context-defining model elements, i.e. `SwcServiceDependency` owned by the `AtomicSwComponentType` 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 `SwcServiceDependency` finally ends up in the one depicted `ramBlock` (colored in blue).

Please note that the graphical representation of the `NvBlockDataMapping` in Figure 11.18 has been simplified for the sake of clarity.

In summary, there is in fact a whole list of motivations that lead to the creation of [constr\_1404]:

- The handling of the dirty-flag within the RTE is done in the context of a `PortPrototype`.
- The currently implemented approach allows for an easy understanding of the relation of data in the application to data in NV-RAM.
- All data in the context of one `SwcServiceDependency` have the identical storage life cycle.
- Jobs triggered in the NvM via operations in the `NvMService` (see [SWS\_NvM\_00734]) interface belong to exactly one `NvBlockDescriptor`.
- As already mentioned, notifications of the NvM towards software components (e.g. `NvMNotifyJobFinished` as described in [SWS\_NvM\_00735] or `NvMNotifyInitBlock`, see [SWS\_NvM\_00736]) belong to exactly one `NvBlockDescriptor`.

**[constr\_1984] Existence of instance reference `NvBlockDataMapping.nvRamBlockElement`** [For each `NvBlockDataMapping`, the instance reference to `ModuleDeclaration` in the role `nvRamBlockElement` shall exist **at the time when the RTE is generated.**]()

Please note that there are use cases for numerous fine-grained (e.g. single bits) data in non-volatile RAM. For this purpose, it would be very inefficient to copy the value of a whole `DataPrototype` that is at least 8 bits long into the non-volatile RAM.

A much better approach is to allow for the mapping of single bits or bit-fields inside a `DataPrototype` in a `PortPrototype` to the `ramBlock` of an `NvBlockDescriptor` directly.

For this purpose the `NvBlockDataMapping` provides the attributes `bitfieldTextTableMaskNvBlockDescriptor` and `bitfieldTextTableMaskPortPrototype`<sup>4</sup>

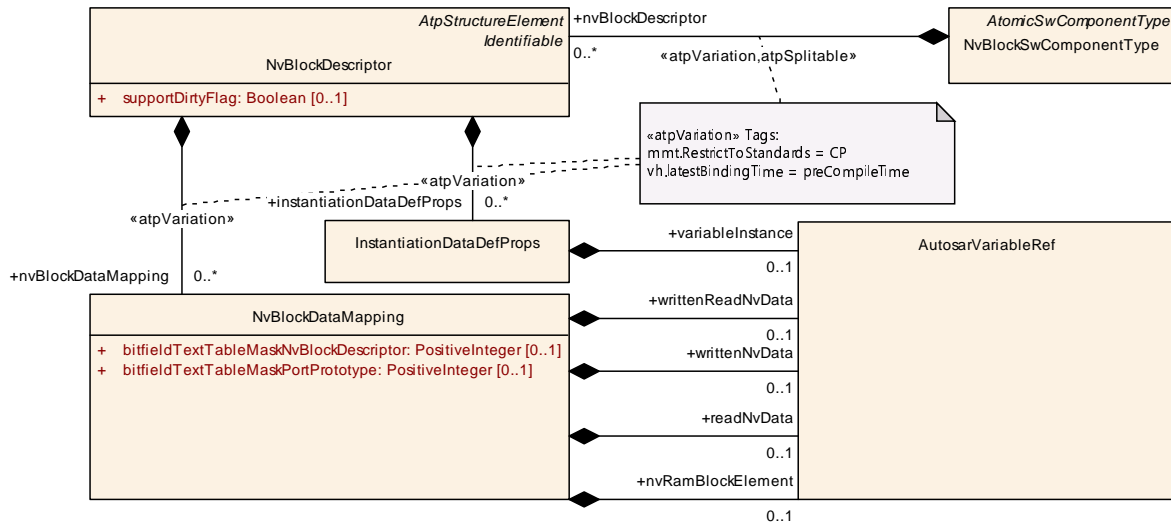


Figure 11.20: Attributes of `NvBlockDataMapping`

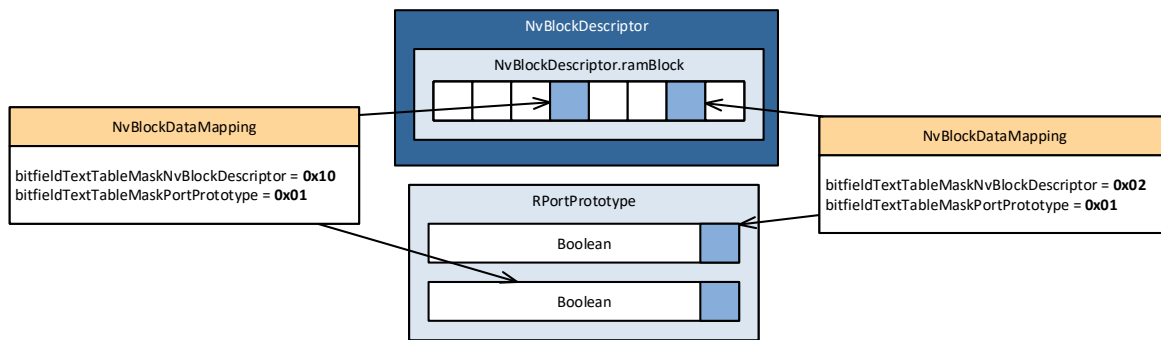


Figure 11.21: Visualization of the approach to map bit-fields inside an `NvBlockSwComponentType`

[TPS\_SWCT\_01799] Mapping of bit-fields between `NvBlockDescriptor` and `PortPrototype` [It is possible to map a bitfield contained in the `DataPrototype` representing the `ramBlock` to a bitfield contained in a `DataPrototype` defined in

<sup>4</sup>The alternative to this approach would have been to shift this "distillation process" into the connection between e.g. an `ApplicationSwComponentType` and an `NvBlockSwComponentType`.

Such an approach comes with considerably heavier side effects and complexity (usage in `DelegationSwConnectors` or as part of a network connection) and is therefore not supported.

the context of a [PortPrototype](#) on the surface of the enclosing [NvBlockSwComponentType](#) by means of the definition of attributes [NvBlockDataMapping.bitfieldTextTableMaskNvBlockDescriptor](#) and [NvBlockDataMapping.bitfieldTextTableMaskPortPrototype](#).

In this case [[constr\\_2013](#)] is not applicable. ]()

The general approach for mapping bit-fields onto each other in the context of an [NvBlockSwComponentType](#) is sketched in Figure 11.21.

Class	NvBlockDataMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::NvBlockComponent			
Note	<p>Defines the mapping between the VariableDataPrototypes in the NvBlockComponents ports and the VariableDataPrototypes of the RAM Block.</p> <p>The data types of the referenced VariableDataPrototypes in the ports and the referenced sub-element (inside a CompositeDataType) of the VariableDataPrototype representing the RAM Block shall be compatible.</p>			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
bitfieldTextTableMaskNvBlockDescriptor	PositiveInteger	0..1	attr	This attribute identifies the applicable bit mask on the side of the Nv Block.
bitfieldTextTableMaskPortPrototype	PositiveInteger	0..1	attr	This attribute identifies the applicable bit mask on the side of the PortPrototype.
nvRamBlockElement	<a href="#">AutosarVariableRef</a>	0..1	aggr	Reference to a VariableDataPrototype of a RAM Block.
readNvData	<a href="#">AutosarVariableRef</a>	0..1	aggr	Reference to a VariableDataPrototype of a pPort of the NvBlockComponent providing read access to the RAM Block. If there is no PortPrototype providing read access (write-only) the reference can be omitted.
writtenNvData	<a href="#">AutosarVariableRef</a>	0..1	aggr	Reference to a VariableDataPrototype of a rPort of the NvBlockComponent providing write access to the RAM Block. If there is no port providing write access (read-only) the reference can be omitted.
writtenReadNvData	<a href="#">AutosarVariableRef</a>	0..1	aggr	Reference to a VariableDataPrototype of a PRPort Prototype of the NvBlockSwComponentType providing write and read access to the RAM Block.

**Table 11.12: NvBlockDataMapping**

### 11.5.5.5 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*. ]()

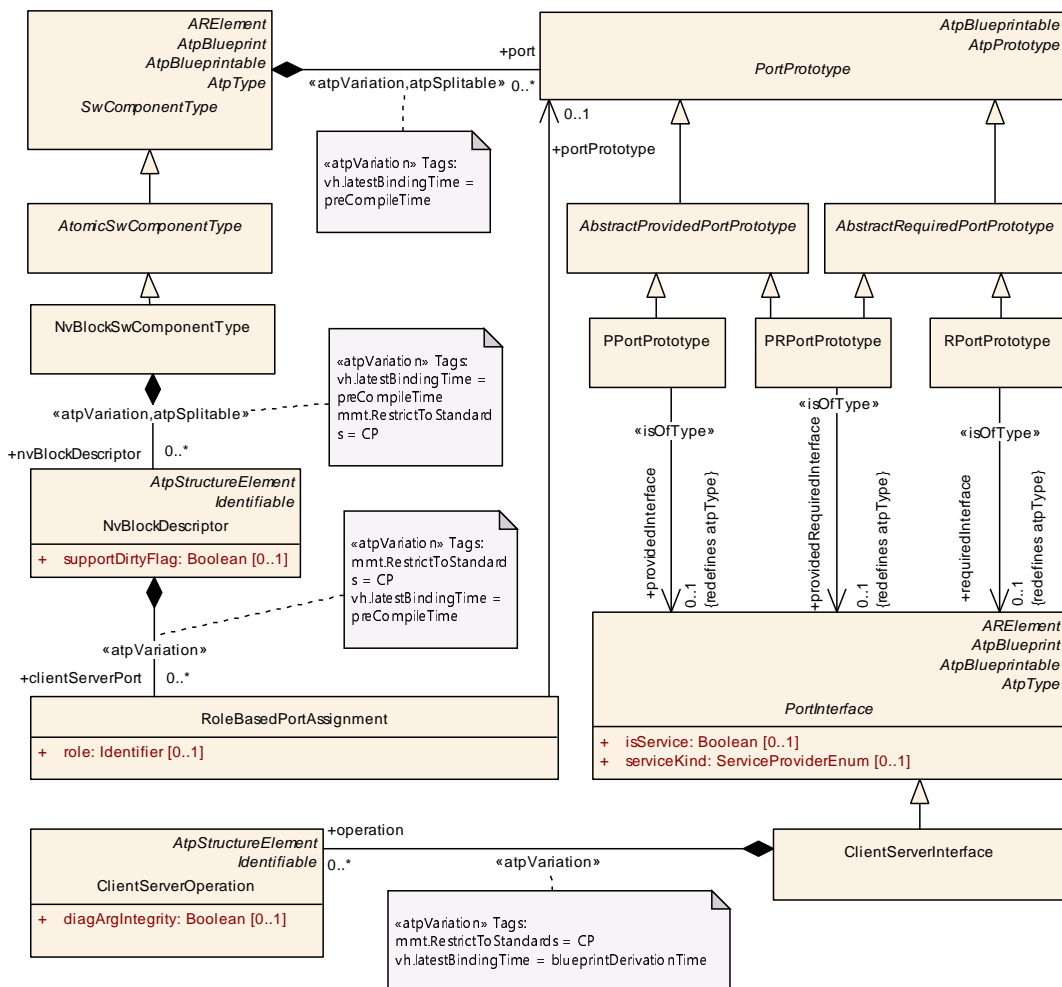
In case of notifications one common callback function is provided by the RTE for each individual kind of notification defined by the attribute `role`.

The `PPortPrototypes` related to a given `NvBlockDescriptor` need to be provided with the same value for a `PortDefinedArgumentValue` in order to make the software work correctly. The provision of the `PortDefinedArgumentValue` is heuristic, but with a further “trick” the reliability of this operation can be much improved.

For all `NvBlockDescriptor.clientServerPort` of a given `NvBlockDescriptor` where attribute `role` is set to the value `NvMService` or `NvMAdmin` collect the `PortPrototype` if it is a `PPortPrototype`. The resulting collection of `PortPrototypes` need to be provided with the same value (of `PortDefinedArgumentValue`).

In this case it is no longer necessary to explicitly model the existence of `PortDefinedArgumentValues` for these `PortPrototypes` since the applicable id value of the NvM can be derived by RTE configuration.

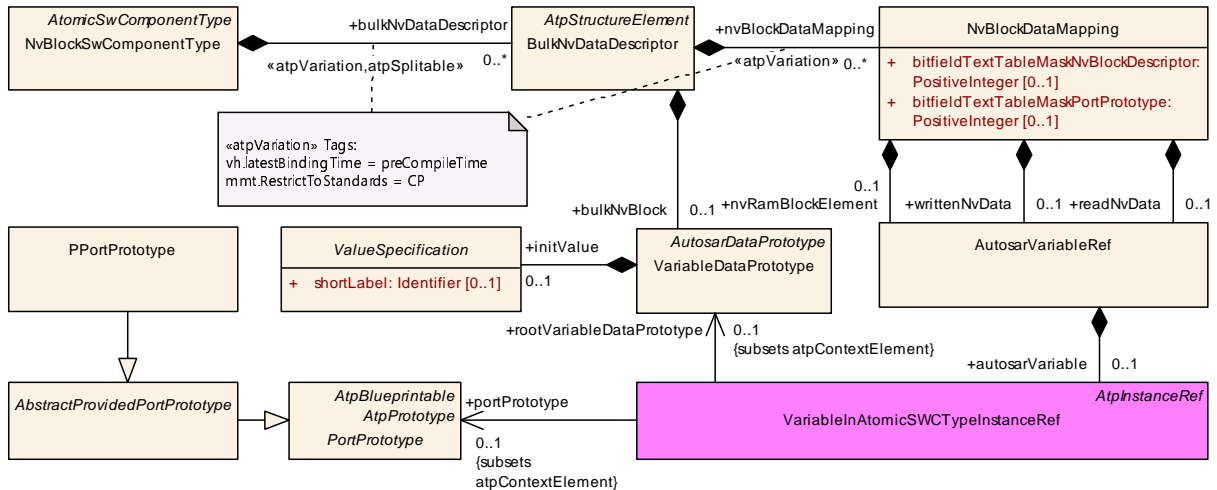
To make this approach work the role value needs to be standardized, see [[constr\\_2014](#)].



**Figure 11.22: NvBlockNotification**

### 11.5.6 BulkNvDataDescriptor

There is a strong use case where application software gets read-only access to data of significant size located in non-volatile memory. The actual data is created or updated e.g. at the end of the production line or maybe during maintenance in a garage.



**Figure 11.23: NvBlockSwComponentType and BulkNvDataDescriptor**

If such a scenario were to be implemented using an **NvBlockDescriptor** then the creation of a RAM mirror of the data in the non-volatile memory would be obligatory. But as described before, the existence of a **RAM mirror would not be necessary at all** in this use case and would only waste a significant amount of precious resources.

It is therefore necessary to be clearly separate the use case for utilizing an **NvBlockDescriptor** from the direct read-only access to non-volatile memory in the meta-model.

For this purpose, model support is provided by the existence of meta-class **BulkNvDataDescriptor**, aggregated at **NvBlockSwComponentType** in the role **bulkNvDataDescriptor**. This modeling is depicted in Figure 11.23.

**[TPS\_SWCT\_01805] Semantics of the aggregation NvBlockSwComponentType.bulkNvDataDescriptor** [The existence of an aggregation at meta-class **NvBlockSwComponentType** in the role **bulkNvDataDescriptor** indicates that read-only access to a block of data in non-volatile memory shall be provided.

Access to this block of data is **provided directly from non-volatile memory** as opposed to via a RAM mirror.]([RS\\_SWCT\\_03225](#))

**[TPS\_SWCT\_01806] Simultaneous aggregation of NvBlockSwComponentType.bulkNvDataDescriptor and NvBlockSwComponentType.nvBlockDescriptor** [The simultaneous existence of aggregations in the roles **bulkNvDataDescriptor** and **nvBlockDescriptor** in the context of the same **NvBlockSwComponentType** is positively supported.]([RS\\_SWCT\\_03225](#))

[TPS\_SWCT\_01807] **Application of `NvBlockDataMapping` on `BulkNvDataDescriptor`** [Via the aggregation of `NvBlockDataMapping` in the role `nvBlockDataMapping` it is possible to make the content of the `BulkNvDataDescriptor` available to a `PPortPrototype` on the surface of the enclosing `NvBlockSwComponentType`.

It is also supported to make only a subset of the content of the `BulkNvDataDescriptor` available at the `PPortPrototype`.] (*RS\_SWCT\_03225*)

[**constr\_1735**] **Limitation of the aggregation of `AutosarVariableRef` in the context of an `NvBlockDataMapping` owned by a `BulkNvDataDescriptor`** [Any `NvBlockDataMapping` owned by a `BulkNvDataDescriptor` shall only aggregate an `AutosarVariableRef` in the role `readNvData` and `nvRamBlockElement` (that in turn refers to the `BulkNvDataDescriptor.bulkNvBlock`) **at the time when the RTE is generated.**]()

Of course, [**constr\_1285**] also applies for the usage of `BulkNvDataDescriptor`.

<b>Class</b>	<b>BulkNvDataDescriptor</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::NvBlockComponent			
<b>Note</b>	This meta-class represents one bulk NV Data Block that is read-only for the application software. The purpose of a bulk NV Data Block is to provide access to information uploaded to the vehicle at e.g. the end of the production line.			
<b>Base</b>	<i>ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
bulkNvBlock	<code>VariableDataPrototype</code>	0..1	aggr	This aggregation represents the actual bulk NVBlock.
nvBlockDataMapping	<code>NvBlockDataMapping</code>	*	aggr	Defines the mapping between the <code>VariableDataPrototypes</code> in the <code>NvBlockComponents</code> ports and the <code>VariableDataPrototypes</code> of the non-volatile memory.  The aggregation of <code>NvBlockDataMapping</code> is subject to variability with the purpose to support the conditional existence of nv data ports.  <b>Stereotypes:</b> <code>atpVariation</code> <b>Tags:</b> <code>vh.latestBindingTime=preCompileTime</code>

**Table 11.13: BulkNvDataDescriptor**

### 11.5.7 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 [\[TPS\\_SWCT\\_01584\]](#).()

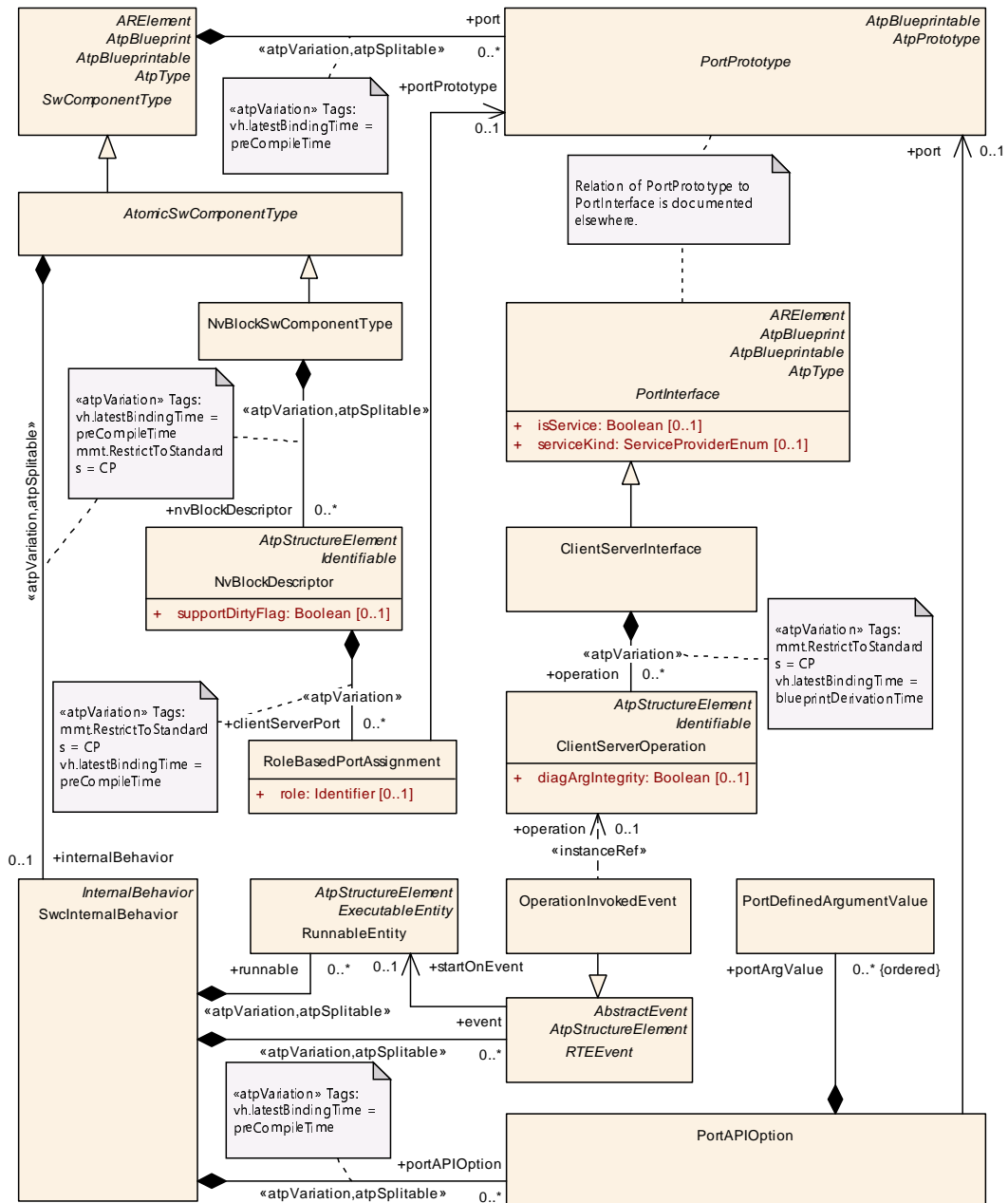


Figure 11.24: **NvBlockSwComponentType** and **SwcInternalBehavior**

[\[TPS\\_SWCT\\_01152\]](#) **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\_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`
- `RunnableEntitys`

](RS\_SWCT\_03225)

**[TPS\_SWCT\_01151] RunnableEntitys do not have further attributes** [The same condition exists for the `RunnableEntitys` 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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

For example, `RunnableEntity.symbol` owned by an `NvBlockSwComponentType` could rightfully be set to `NvM_ReadBlock` [35] but an arbitrary value like `ReadThisBlock` is not permitted.

The rationale for [constr\_1234] is that the `RunnableEntitys` 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 [6].

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`
- `RunnableEntities` triggered by `OperationInvokedEvents` (server `RunnableEntities`)
- `RunnableEntities` 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`
- `RunnableEntities` triggered by `TimingEvents`
- `RunnableEntities` triggered by `DataReceivedEvents`
- `RunnableEntities` triggered by `SwcModeSwitchEvents`
- `DataTypeMappingSet`

This rule shall be imposed **at the time when the RTE is generated.**]()

**[constr\_1309] Existence of `NvBlockDescriptor.timingEvent`** [The attribute `NvBlockDescriptor.timingEvent` shall exist **at the time when the RTE is generated** 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*)

**[TPS\_SWCT\_01662] Applicability of `DataTypeMappingSets` inside an `NvBlock-SwComponentType`** [The `DataTypeMappingSets` to be applied for a given `NvBlockDescriptor` is the superset of `NvBlockDescriptor.dataTypeMapping` and `InternalBehavior.dataTypeMapping`.](*RS\_SWCT\_03225*)

## 12 Software Component Documentation

AUTOSAR supports documentation of software component types by adopting the principles of ASAM-FSX [36] 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 [11].

**[TPS\_SWCT\_01062] Documentation of software-components** [The documentation of a `SwComponentType` is composed of several `Chapters`.

Some `Chapters` are predefined, describing the component from the perspective of different activities performed on the component, like:

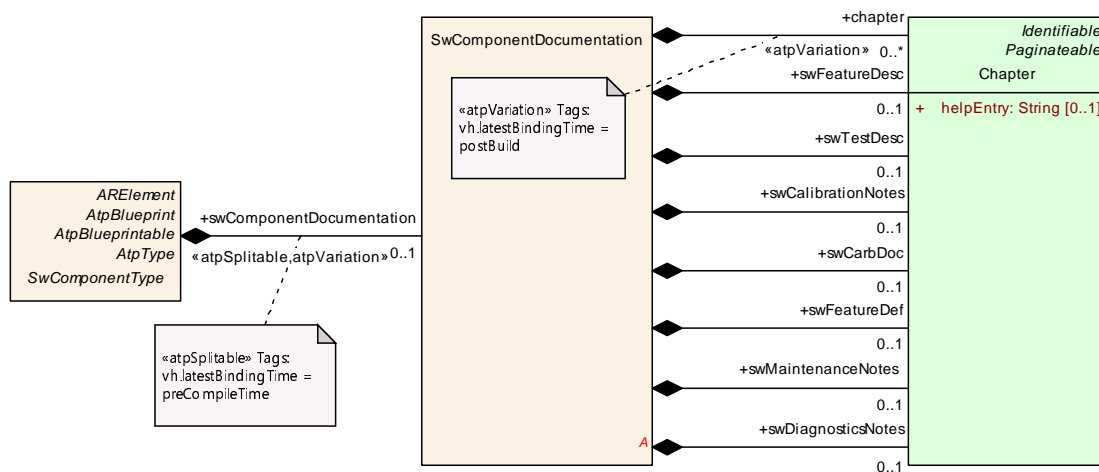
- testing it (`swTestDesc`)
- maintaining it (`swMaintenanceNotes`)
- calibrating it (`swCalibrationNotes`)
- performing diagnostic (`swDiagnosticsNotes`)

](`RS_SWCT_02110`, `RS_SWCT_03230`)

The documentation of a `SwComponentType` is shown in figure 12.1.

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.



**Figure 12.1: Software component documentation**

The predefined `Chapters` typically provide informal guideline (e.g. recommendation) or documentation.

Formal information can be captured using special data groups [11] 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 `<<atpSplittable>>` from the component).

Each of the predefined and free [Chapters](#) follows the `<<atpVariation>>` stereotype to support variant handling (see [11]) on the documentation at the [Chapter](#) level.

These [VariationPoints](#) set the latest binding time to the value `AdditionalBindingTimeEnum.postBuild` 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.

<b>Class</b>	<b>SwComponentDocumentation</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SoftwareComponentDocumentation			
<b>Note</b>	This class specifies the ability to write dedicated documentation to a component type according to ASAM FSX.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
chapter	<a href="#">Chapter</a>	*	aggr	<p>These chapters provide additional information about the software component that do not fit in the other chapters.</p> <p>Note that this is subject to variation because Chapter aggregations in the role chapter are variant within the documentation in general.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b>            vh.latestBindingTime=postBuild            xml.roleElement=true            xml.roleWrapperElement=false            xml.sequenceOffset=100            xml.typeElement=false</p>
swCalibration Notes	<a href="#">Chapter</a>	0..1	aggr	<p>This element contains calibration instructions and hints for a calibration engineer.</p> <p><b>Tags:</b>            xml.roleElement=true            xml.sequenceOffset=60            xml.typeElement=false</p>
swCarbDoc	<a href="#">Chapter</a>	0..1	aggr	<p>This element records the documentation requested by CARB.</p> <p><b>Tags:</b>            xml.roleElement=true            xml.sequenceOffset=80            xml.typeElement=false</p>
swDiagnostics Notes	<a href="#">Chapter</a>	0..1	aggr	<p>This element contains general information about diagnostics issues within the component.</p> <p><b>Tags:</b>            xml.roleElement=true            xml.sequenceOffset=75            xml.typeElement=false</p>





Class	SwComponentDocumentation			
swFeatureDef	<a href="#">Chapter</a>	0..1	aggr	<p>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.</p> <p><b>Tags:</b> xml.roleElement=true xml.sequenceOffset=20 xml.typeElement=false</p>
swFeatureDesc	<a href="#">Chapter</a>	0..1	aggr	<p>This element contains the textual description of the software functionality of this software component. Expert should write this description.</p> <p><b>Tags:</b> xml.roleElement=true xml.sequenceOffset=30 xml.typeElement=false</p>
swMaintenance Notes	<a href="#">Chapter</a>	0..1	aggr	<p>This element contains information regarding the software maintenance of the component.</p> <p><b>Tags:</b> xml.roleElement=true xml.sequenceOffset=70 xml.typeElement=false</p>
swTestDesc	<a href="#">Chapter</a>	0..1	aggr	<p>This element contains suggestions and hints for the test of the software functionality of this software component.</p> <p><b>Tags:</b> xml.roleElement=true xml.sequenceOffset=50 xml.typeElement=false</p>

**Table 12.1: SwComponentDocumentation**

Class	Chapter			
Package	M2::MSR::Documentation::Chapters			
Note	This meta-class represents a chapter of a document. Chapters are the primary structuring element in documentation.			
Base	<i>ARObject</i> , <i>DocumentViewSelectable</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Paginateable</i> , <i>Referrable</i>			
Attribute	Type	Mult.	Kind	Note
chapterModel	ChapterModel	1	aggr	<p>This represents the overall contents of the chapter.</p> <p><b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.typeElement=false xml.typeWrapperElement=false</p>
helpEntry	<a href="#">String</a>	0..1	attr	<p>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.</p> <p>Maybe it is a concatenated Identifier, but as of now we leave it as an arbitrary string.</p> <p><b>Tags:</b>xml.attribute=true</p>

**Table 12.2: Chapter**

<b>Enumeration</b>	<b>AdditionalBindingTimeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling
<b>Note</b>	This enumeration specifies the additional binding times applicable for vh.latestBindingTime of variation points.
<b>Literal</b>	<b>Description</b>
blueprintDerivationTime	The point in time when an object is created from a blueprint. <b>Tags:</b> atp.EnumerationLiteralIndex=0
postBuild	After the executable has been built. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 12.3: AdditionalBindingTimeEnum**

## 13 Service Dependencies and Service Use Cases

### 13.1 Overview

Meta-class `SwcServiceDependency` represents a powerful concept to describe the service-related capabilities of an `AtomicSwComponentType`.

It is still required to understand how to configure `SwcServiceDependency` and related meta-classes for specific service use cases.

This chapter contains a detailed description of the meta-classes related to `SwcServiceDependency` in the context of specific service use cases, as well as modeling hints for the configuration of the respective service use cases.

### 13.2 NvM Service Dependencies

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 [35]).

A diagram that explains the modeling of `ServiceNeeds` for the NVRAM Manager Service can be found in Figure 11.16.

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.

<b>Enumeration</b>	<b>RamBlockStatusControlEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::NvBlockComponent
<b>Note</b>	This enumeration type defines options for how the management of the ramBlock status is controlled.
<b>Literal</b>	<b>Description</b>
api	The ramBlock status is controlled via service interface by usage of the SetRamBlockStatus operation. <b>Tags:</b> atp.EnumerationLiteralIndex=0
nvRamManager	The ramBlock status is controlled exclusively by the Nv Ram Manager. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 13.1: RamBlockStatusControlEnum**



### 13.2.1 Nvm Use Case: Permanent RAM Block

Scenario: a `AtomicSwComponentType` is using 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

]([RS\\_SWCT\\_03225](#))

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.

### 13.2.2 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 `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` [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 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

](RS\_SWCT\_03225)

**[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`.

This rule shall be imposed **at the time when the RTE is generated.** ]()

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].

### 13.2.3 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

]([RS\\_SWCT\\_03225](#))

For more information please refer to [\[SWS\\_NvM\\_00734\]](#), [\[SWS\\_NvM\\_00735\]](#), [\[SWS\\_NvM\\_00736\]](#), [\[SWS\\_NvM\\_00737\]](#), and [\[SWS\\_NvM\\_00738\]](#).

## 13.2.4 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 supporting this use case the value of attribute `SwcServiceDependency.category` shall be set to `NV_BLOCK_COMPONENT`.] ([RS\\_SWCT\\_03225](#))

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 11.5.5) is not in the scope of this use case.

## 13.3 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 [37].

### 13.3.1 Watchdog Service use Case: Local Supervision

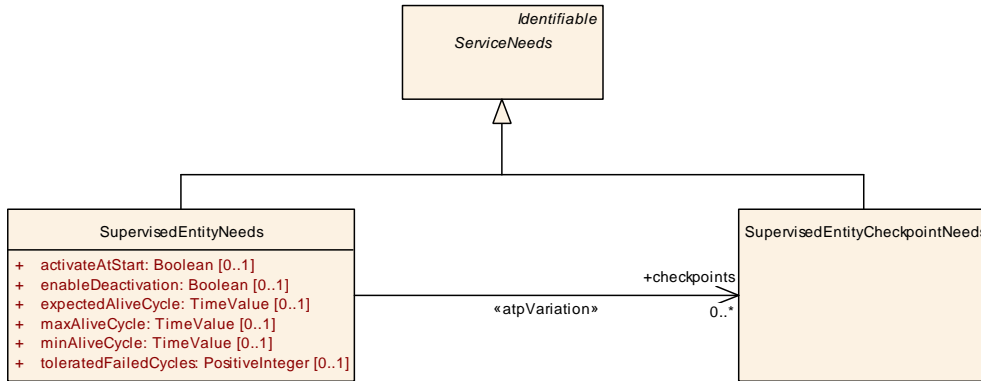
The service interaction with the *Watchdog Manager* consists of two aspects:

- supervised entity
- checkpoint

For each of the two aspects a separated `ServiceNeeds` is defined. However, the `SwcServiceDependencies` that own these `ServiceNeeds` are semantically bound and cannot be used independently of each other.

In other words, the usage of two kinds of `SwcServiceDependency` in concert creates a higher-level semantics. Of course, in order to express this higher-level semantics a reference between the `SwcServiceDependencies` has to be available.

However, since the `SwcServiceDependency` represents a generic concept the actual reference needs to be implemented on the level of specific subclass of `ServiceNeeds`, in this case the `SupervisedEntityNeeds` and the `SupervisedEntityCheckpointNeeds`.



**Figure 13.1: Modeling of `ServiceNeeds` for the watchdog**

The former refers to the latter in order to express the relation of a supervised entity to its checkpoints.

<b>Class</b>	<b>SupervisedEntityNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the abstract needs on the configuration of the Watchdog Manager for one specific Supervised Entity.			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
activateAtStart	Boolean	0..1	attr	True/false: supervision activation status of Supervised Entity shall be enabled/disabled at start.
checkpoints	SupervisedEntityCheckpointNeeds	*	ref	This reference indicates the checkpoints belonging to the Supervised Entity. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
enable Deactivation	Boolean	0..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
expectedAlive Cycle	TimeValue	0..1	attr	Expected cycle time of alive trigger of this Supervised Entity (in seconds).
maxAliveCycle	TimeValue	0..1	attr	Maximum cycle time of alive trigger of this Supervised Entity (in seconds).
minAliveCycle	TimeValue	0..1	attr	Minimum cycle time of alive trigger of this Supervised Entity (in seconds).
toleratedFailed Cycles	PositiveInteger	0..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 13.2: SupervisedEntityNeeds**

**[constr\_1985] Existence of the reference `SupervisedEntityNeeds.toleratedFailedCycles`** [For each `SupervisedEntityNeeds`, the reference to `BswInternalBehavior` in the role `toleratedFailedCycles` shall exist **at the time when the RTE is generated.** ]()

<b>Class</b>	<b>SupervisedEntityCheckpointNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the abstract needs on the configuration of the Watchdog Manager to support a Checkpoint for a Supervised Entity.			
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.3: SupervisedEntityCheckpointNeeds**

### 13.3.2 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

] ([RS\\_SWCT\\_00030](#))

For more information please refer to [SWS\_WdgM\_00336].

### 13.3.3 Watchdog Service use Case: Control global supervision or get global supervision status

Scenario: an `AtomicSwComponentType` either controls the global operation of the watchdog manager or gets information about the current operation status, requiring at least one of the following use cases:

- Set the current mode of Watchdog Manager
- Get the current mode of the Watchdog Manager

- Get the global supervision status of the Watchdog Manager
- Identifier of the supervised entity that first reached the expired state
- Instruct the Watchdog Manager to cause a watchdog reset

For instance, the software-component sets the current mode of the Watchdog Manager according to the operational state of the ECU or polls the global supervision status.

In this case the following setup applies:

**[TPS\_SWCT\_01703] Setup for `AtomicSwComponentType` which sets or gets the Global Supervision Status** [

**ServiceNeeds kind** `GlobalSupervisionNeeds`

**RoleBasedPortAssignment** valid roles:

- `WdgM_GlobalSupervision` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](`RS_SWCT_00030`)

<b>Class</b>	<code>GlobalSupervisionNeeds</code>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the abstract needs on the configuration of the Watchdog Manager to get access on the Global Supervision control and status interface.			
<b>Base</b>	<code>ARObject</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code> , <code>ServiceNeeds</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.4: GlobalSupervisionNeeds**

### 13.3.4 Watchdog Service use Case: Software-component wants to obtain the status of a local supervision

Scenario: a software-component wants to obtain the status of a local supervision. For this purpose, it exposes an `RPortPrototype` typed by `ClientServerInterface` `WdgM_LocalSupervisionStatus` towards the WdgM.

**[TPS\_SWCT\_01704] Definition of supervised entity** [

**ServiceNeeds kind** : `SupervisedEntityNeeds`

**RoleBasedPortAssignment** valid roles:

- `WdgM_LocalSupervisionStatus` [1]



**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

Please note that an `SwcInternalBehavior` may provide several `SupervisedEntityNeeds` elements where each defines the requirements in relation to one supervised entity.

Note that in this situation an `AtomicSwComponentType` contains several *Checkpoints* that refer to a *Supervised Entity*.

In this case it is required that the *Supervised Entity* indicates to the *Watchdog Manager* the existence this *Checkpoint* for configuration and at runtime that the *Supervised Entity* has reached the *Checkpoint*.

For more information please refer to [SWS\_WdgM\_91004]

**13.3.5 Watchdog Service use Case: Software-component wants to report a checkpoint**

Scenario: A software-component want to report a checkpoint to the WdgM. For this purpose the software-component exposes an `RPortPrototype` to the WdgM.

**[TPS\_SWCT\_01705] Definition of Checkpoints [**

**ServiceNeeds kind** : `SupervisedEntityCheckpointNeeds`

**RoleBasedPortAssignment** valid roles:

- `WdgM_LocalSupervision [1]`

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

Please note that this scenario covers the reporting of a single checkpoint. In many cases, several checkpoints are reported in the context of a given supervised entity.

For more information please refer to [SWS\_WdgM\_00333].

## 13.4 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”.

<b>Class</b>	<b>ComMgrUserNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the abstract needs on the configuration of the Communication Manager for one "user".			
<b>Base</b>	<code>ARObject</code> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
maxComm Mode	MaxCommModeEnum	0..1	attr	Maximum communication mode requested by this ComM user.

**Table 13.5: ComMgrUserNeeds**

### 13.4.1 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

**]([RS\\_SWCT\\_03200](#))**

For more information please refer to [SWS\_ComM\_00847].

### 13.4.2 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]

]([RS\\_SWCT\\_03200](#))

For more information please refer to [SWS\_ComM\_00848].

### 13.4.3 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

]([RS\\_SWCT\\_03200](#))

For more information please refer to [SWS\_ComM\_00741].

### 13.4.4 Read PNC ComM Mode

Scenario: an [AtomicSwComponentType](#) reads the current PNC ComM mode.

[TPS\_SWCT\_01811] [AtomicSwComponentType](#) reads the current PNC ComM mode [

**ServiceNeeds kind** : [ComMgrUserNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [ComM\\_UserRequest](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroup**

Reference to the applicable [PortGroup](#) associated with the particular partial network.

]([RS\\_SWCT\\_03200](#))

For clarification, the difference between the two service use case descriptions [TPS\_SWCT\_01019] and [TPS\_SWCT\_01811] is that the latter reads the ComM mode in the context of a PNC definition while the former does not consider partial networking.

## 13.5 ECU State Manager Service Needs

The meta-class [EcuStateMgrUserNeeds](#) is used to define the requirements to configure the ECU State Manager Service.

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 [38]).

<b>Class</b>	<b>EcuStateMgrUserNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.6: EcuStateMgrUserNeeds**

### 13.5.1 EcuM 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 [

**RoleBasedPortAssignment** valid roles:

- EcuM\_ShutdownTarget [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroup**

N/A

](RS\_SWCT\_03200)

### 13.5.2 EcuM 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 [

**RoleBasedPortAssignment** valid roles:

- EcuM\_BootTarget [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroup**

N/A

](RS\_SWCT\_03200)

### 13.5.3 EcuM 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 [

**RoleBasedPortAssignment** valid roles:

- EcuM\_AlarmClock [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroup**

N/A

]([RS\\_SWCT\\_03200](#))

## 13.6 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`.

<b>Class</b>	<b>BswMgrNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the abstract needs on the configuration of the Basic Software Manager for one "user".			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.7: BswMgrNeeds**

### 13.6.1 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\\_03200](#), [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*.

### 13.6.2 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\_00200]).

In this case the following rules apply:

**[TPS\_SWCT\_01552] Software-component acts as a mode manager** [

**ServiceNeeds** kind `BswMgrNeeds`

**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`.

**[TPS\_SWCT\_01664] BswM acts as a mode requester towards an application mode manager** [The `SwcServiceDependency` that covers this use case shall refer to an `RPortPrototype` for the reception of the mode request and optionally to a `PPortPrototype` for the sending of the mode switch notification.

**ServiceNeeds kind** `BswMgrNeeds`

**RoleBasedPortAssignment** valid roles:

- `AppModeInterface` [0..1]

**RoleBasedDataAssignment** valid roles:

- `AppModeRequestInterface` [1]

**RepresentedPortGroup**

none

]([RS\\_SWCT\\_03110](#), [RS\\_SWCT\\_03200](#), [RS\\_SWCT\\_03203](#))



**[constr\_1680] Existence of attribute `RoleBasedDataAssignment.usedDataElement.localVariable` for `RoleBasedDataAssignment.role = AppModeRequestInterface`** [If the attribute `RoleBasedDataAssignment.role` is set to the value `AppModeRequestInterface` then the reference `RoleBasedDataAssignment.usedDataElement.localVariable` shall not exist **at the time when the RTE is generated.** ]()

For explanation of the existence of [constr\_1680], it is not intended to provide access to local variables inside the `SwcInternalBehavior`.

### 13.6.3 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\_00202]).

In this case the following rules apply:

**[TPS\_SWCT\_01553] Software-component acts as a mode user** [

**ServiceNeeds** kind `BswMgrNeeds`

**RoleBasedPortAssignment** valid roles:

- `AppModeInterface` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroup**

none

](*RS\_SWCT\_03110, RS\_SWCT\_03200, RS\_SWCT\_03203*)

### 13.6.4 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\_00201]).

In this case the following rules apply:

[TPS\_SWCT\_01554] Software-component acts as a mode requester [

ServiceNeeds kind [BswMgrNeeds](#)

RoleBasedPortAssignment

N/A

RoleBasedDataAssignment valid roles:

- `AppModeRequestInterface` [1]

RepresentedPortGroup

none

]([RS\\_SWCT\\_03110](#), [RS\\_SWCT\\_03200](#), [RS\\_SWCT\\_03202](#))

Please note that [[constr\\_1680](#)] applies for this service use case.

## 13.7 Crypto Service Dependencies

### 13.7.1 Overview

The meta-classes [CryptoServiceNeeds](#) and [CryptoServiceJobNeeds](#) are used to define requirements for the configuration of the `CryptoServiceManager` respectively the crypto stack.

The usage of meta-class [CryptoKeyManagementNeeds](#) is described in section [13.7.4](#).

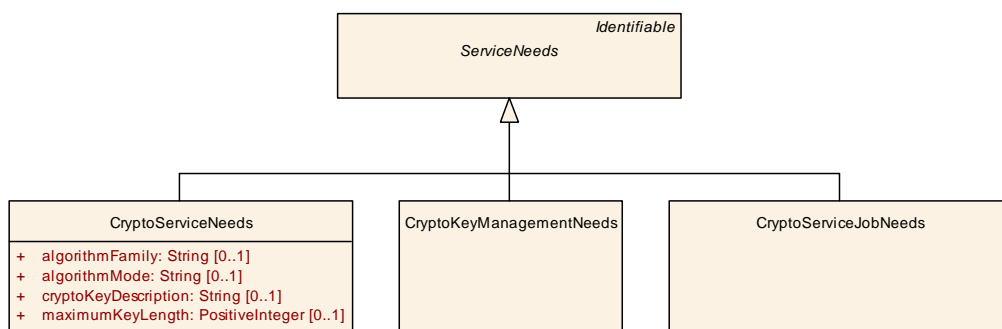
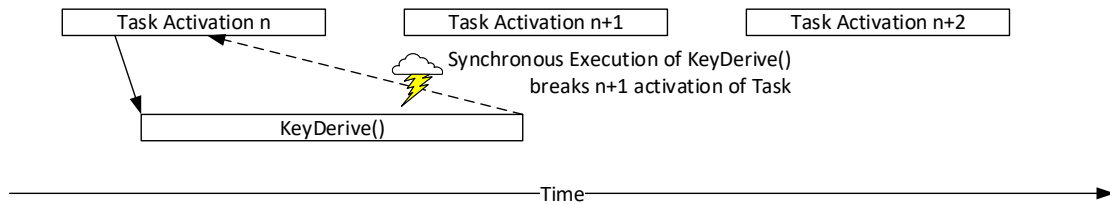


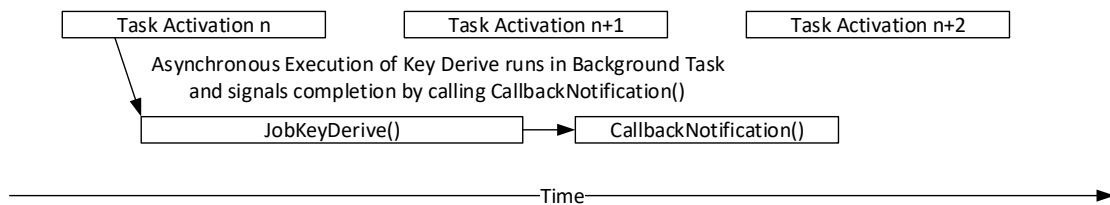
Figure 13.2: Modeling of subclasses of [ServiceNeeds](#) for cryptographic use cases

Please note that there are cryptographic APIs that build upon the creation of jobs that run asynchronously. The reason for this policy is that cryptographic operations - in many cases by design - tend to run for comparatively long time for each call. This behavior is visualized in [Figure 13.3](#).



**Figure 13.3: Cryptographic operation that requires a too-long execution time**

Execution of these operations synchronously in the main function would block the respective module intolerably and therefore the job API is an important measure to keep the execution of software manageable. This behavior is visualized in Figure 13.4.



**Figure 13.4: Execution of cryptographic operation using the job API**

It is important to note that the asynchronous character of the execution is implemented on the server side and has nothing to do with asynchronous calling behavior on the client side (for more explanation about client-side calling behavior, please refer to section 7.5.2.1).

<b>Class</b>	<b>CryptoServiceNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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 a software-component belong to this ConfigID.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
algorithmFamily	<a href="#">String</a>	0..1	attr	This attribute represents a description of the family (e.g. AES) of crypto algorithm implemented by the crypto service use case.
algorithmMode	<a href="#">String</a>	0..1	attr	This meta-class has the ability to represent a crypto service use case.
cryptoKey Description	<a href="#">String</a>	0..1	attr	This attribute allows for a verbal description of the applicable cryptographic key. The goal is to pass a hint for the integrator about how to treat the corresponding service use case.
maximumKey Length	PositiveInteger	0..1	attr	The maximum length of a cryptographic key, that is used by the software-component or module for this configuration. Unit: bit.

**Table 13.8: CryptoServiceNeeds**

<b>Class</b>	<b>CryptoServiceJobNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class shall be taken to indicate that the service use case modeled with this kind of Service Needs assumes the usage of the crypto job API.			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.9: CryptoServiceJobNeeds**

[TPS\_SWCT\_01727] Suffix used for the resulting name of the **PortInterface** for crypto **PortInterfaces** [The `_{Config}` or (where applicable) `_{Primitive}` suffix used for the resulting name of the **PortInterface** for the respective crypto service shall be taken from the `shortName` of the applicable `SwcServiceDependency`.]()

## 13.7.2 Crypto Service Use Cases

### 13.7.2.1 Crypto Service Use Case: Hash calculation

Scenario: a `AtomicSwComponentType` uses the hash calculation of the Crypto Service. In this case the following setup applies:

[TPS\_SWCT\_02020] `AtomicSwComponentType` uses the hash calculation of the **Crypto Service** [

**ServiceNeeds kind** : `CryptoServiceNeeds`

**RoleBasedPortAssignment** valid roles:

- `CsmHash [0..1]`
- `CsmHash_{Config} [0..1]`
- `CallbackNotification [0..1]`

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

] (`RS_SWCT_00030`)

Please note that the resolution of the name fragment `{Config}` that appears in the `shortName` of the `ClientServerInterface` `CsmHash_{Config}` is regulated by [TPS\_SWCT\_01727].

### 13.7.2.2 Crypto 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 applies:

**[TPS\_SWCT\_02021] [AtomicSwComponentType](#) uses the message authentication code (MAC) calculation of the Crypto Service [**

**ServiceNeeds kind :** [CryptoServiceNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [CsmMacGenerate](#) [0..1]
- [CsmMacGenerate\\_{Config}](#) [0..1]
- [CallbackNotification](#) [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

Please note that the resolution of the name fragment `{Config}` that appears in the `shortName` of the [ClientServerInterface](#) `CsmMacGenerate_{Config}` is regulated by [\[TPS\\_SWCT\\_01727\]](#).

### 13.7.2.3 Crypto 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 applies:

**[TPS\_SWCT\_02022] [AtomicSwComponentType](#) uses the message authentication code (MAC) verification of the Crypto Service [**

**ServiceNeeds kind :** [CryptoServiceNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [CsmMacVerify](#) [0..1]
- [CsmMacVerify\\_{Config}](#) [0..1]
- [CallbackNotification](#) [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

Please note that the resolution of the name fragment {Config} that appears in the `shortName` of the `ClientServerInterface CsmMacVerify_{Config}` is regulated by [TPS\_SWCT\_01727].

#### 13.7.2.4 Crypto 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 applies:

**[TPS\_SWCT\_02024] `AtomicSwComponentType` uses the generation of random numbers of the Crypto Service [**

**ServiceNeeds kind :** `CryptoServiceNeeds`

**RoleBasedPortAssignment** valid roles:

- `CsmRandomGenerate` [0..1]
- `CsmRandomGenerate_{Config}` [0..1]
- `CallbackNotification` [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

Please note that the resolution of the name fragment {Config} that appears in the `shortName` of the `ClientServerInterface CsmRandomGenerate_{Config}` is regulated by [TPS\_SWCT\_01727].

#### 13.7.2.5 Crypto Service Use Case: Encryption with Authenticated Encryption with Associated Data (AEAD)

Scenario: a `AtomicSwComponentType` uses the Authenticated Encryption with Associated Data (AEAD) of the Crypto Service. In this case the following setup applies:

**[TPS\_SWCT\_02025] `AtomicSwComponentType` uses the Authenticated Encryption with Associated Data (AEAD) of the Crypto Service [**

**ServiceNeeds kind :** `CryptoServiceNeeds`

**RoleBasedPortAssignment** valid roles:

- `CsmAEADEncrypt` [0..1]

- CsmAEADEncrypt\_{Config} [0..1]
- CallbackNotification [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

|(RS\_SWCT\_00030)

Please note that the resolution of the name fragment {Config} that appears in the `shortName` of the `ClientServerInterface` CsmAEADEncrypt\_{Config} shall be resolved according to [TPS\_SWCT\_01727].

**13.7.2.6 Crypto Service Use Case: Decryption with Authenticated Encryption with Associated Data (AEAD)**

Scenario: a `AtomicSwComponentType` uses the Authenticated Encryption with Associated Data (AEAD) of the Crypto Service. In this case the following setup applies:

[TPS\_SWCT\_02026] **AtomicSwComponentType uses the Authenticated Encryption with Associated Data (AEAD) of the Crypto Service** [

**ServiceNeeds kind** : `CryptoServiceNeeds`

**RoleBasedPortAssignment** valid roles:

- CsmAEADDecrypt [0..1]
- CsmAEADDecrypt\_{Config} [0..1]
- CallbackNotification [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

|(RS\_SWCT\_00030)

Please note that the resolution of the name fragment {Config} that appears in the `shortName` of the `ClientServerInterface` CsmAEADDecrypt\_{Config} is regulated by [TPS\_SWCT\_01727].

**13.7.2.7 Crypto Service Use Case: encryption**

Scenario: a `AtomicSwComponentType` uses the encryption of the Crypto Service. In this case the following setup applies:

[TPS\_SWCT\_02027] **AtomicSwComponentType** uses the encryption of the Crypto Service [

**ServiceNeeds kind** : *CryptoServiceNeeds*

**RoleBasedPortAssignment** valid roles:

- *CsmEncrypt* [0..1]
- *CsmEncrypt\_{Config}* [0..1]
- *CallbackNotification* [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](*RS\_SWCT\_00030*)

Please note that the resolution of the name fragment {Config} that appears in the *shortName* of the *ClientServerInterface* *CsmEncrypt\_{Config}* is regulated by [TPS\_SWCT\_01727].

### 13.7.2.8 Crypto Service Use Case: decryption

Scenario: a *AtomicSwComponentType* uses the decryption of the Crypto Service. In this case the following setup applies:

[TPS\_SWCT\_02028] **AtomicSwComponentType** uses the decryption of the Crypto Service [

**ServiceNeeds kind** : *CryptoServiceNeeds*

**RoleBasedPortAssignment** valid roles:

- *CsmDecrypt* [0..1]
- *CsmDecrypt\_{Config}* [0..1]
- *CallbackNotification* [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](*RS\_SWCT\_00030*)

Please note that the resolution of the name fragment {Config} that appears in the *shortName* of the *ClientServerInterface* *CsmDecrypt\_{Config}* is regulated by [TPS\_SWCT\_01727].



### 13.7.2.9 Crypto Service Use Case: signature generation

Scenario: a [AtomicSwComponentType](#) uses the signature generation of the Crypto Service. In this case the following setup applies:

**[TPS\_SWCT\_02031] [AtomicSwComponentType](#) uses the signature generation of the Crypto Service** [

**ServiceNeeds kind** : [CryptoServiceNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [CsmSignatureGenerate](#) [0..1]
- [CsmSignatureGenerate\\_{Config}](#) [0..1]
- [CallbackNotification](#) [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

Please note that the resolution of the name fragment `{Config}` that appears in the `shortName` of the [ClientServerInterface](#) `CsmSignatureGenerate_{Config}` is regulated by [\[TPS\\_SWCT\\_01727\]](#).

### 13.7.2.10 Crypto Service Use Case: signature verification

Scenario: a [AtomicSwComponentType](#) uses the signature verification of the Crypto Service. In this case the following setup applies:

**[TPS\_SWCT\_02032] [AtomicSwComponentType](#) uses the signature verification of the Crypto Service** [

**ServiceNeeds kind** : [CryptoServiceNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [CsmSignatureVerify](#) [0..1]
- [CsmSignatureVerify\\_{Config}](#) [0..1]
- [CallbackNotification](#) [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

Please note that the resolution of the name fragment {Config} that appears in the `shortName` of the `ClientServerInterface CsmSignatureVerify_{Config}` is regulated by [TPS\_SWCT\_01727].

### 13.7.2.11 Crypto Service Use Case: usage of key management

Scenario: a `AtomicSwComponentType` uses the key management of the Crypto Service. In this case the following setup applies:

[TPS\_SWCT\_01726] `AtomicSwComponentType` uses the key management of the Crypto Service [

**ServiceNeeds kind** : `CryptoServiceNeeds`

**RoleBasedPortAssignment** valid roles:

- `CsmKeyManagement_{Config}` [1]
- `CallbackNotification` [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

Please note that the resolution of the name fragment {Config} that appears in the `shortName` of the `ClientServerInterface CsmKeyManagement_{Config}` is regulated by [TPS\_SWCT\_01727].

## 13.7.3 Crypto Service Job Use Cases

### 13.7.3.1 Crypto Service Use Case: usage of job API to set key valid

Scenario: a `AtomicSwComponentType` uses the **job API** of the Crypto Service to set a key valid. In this case the following setup applies:

[TPS\_SWCT\_01776] `AtomicSwComponentType` uses the API of the Crypto Service to set a key valid [

**ServiceNeeds kind** : `CryptoServiceJobNeeds`

**RoleBasedPortAssignment** valid roles:

- `CsmJobKeySetValid_{Primitive}` [1]
- `CallbackNotification` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

Please note that the resolution of the name fragment {Primitive} that appears in the `shortName` of the `ClientServerInterface` `CsmJobKeySetValid_{Primitive}` is regulated by [TPS\_SWCT\_01727].

**13.7.3.2 Crypto Service Use Case: usage of job API to create a random seed**

Scenario: a `AtomicSwComponentType` uses the **job API** of the Crypto Service to create a random seed. In this case the following setup applies:

[TPS\_SWCT\_01777] `AtomicSwComponentType` uses the API of the Crypto Service to create a random seed [

**ServiceNeeds kind** : `CryptoServiceJobNeeds`

**RoleBasedPortAssignment** valid roles:

- `CsmJobRandomSeed_{Primitive}` [1]
- `CallbackNotification` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

Please note that the resolution of the name fragment {Primitive} that appears in the `shortName` of the `ClientServerInterface` `CsmJobRandomSeed_{Primitive}` is regulated by [TPS\_SWCT\_01727].

**13.7.3.3 Crypto Service Use Case: usage of job API to generate a key**

Scenario: a `AtomicSwComponentType` uses the **job API** of the Crypto Service to generate a key. In this case the following setup applies:

[TPS\_SWCT\_01778] `AtomicSwComponentType` uses the API of the Crypto Service to generate a key [

**ServiceNeeds kind** : `CryptoServiceJobNeeds`

**RoleBasedPortAssignment** valid roles:

- CsmJobKeyGenerate\_{Primitive} [1]
- CallbackNotification [1]

#### RoleBasedDataAssignment

N/A

#### RepresentedPortGroups

N/A

]([RS\\_SWCT\\_00030](#))

Please note that the resolution of the name fragment {Primitive} that appears in the `shortName` of the `ClientServerInterface` `CsmJobKeyGenerate_{Primitive}` is regulated by [[TPS\\_SWCT\\_01727](#)].

### 13.7.3.4 Crypto Service Use Case: usage of job API to derive a key

Scenario: a `AtomicSwComponentType` uses the **job API** of the Crypto Service to derive a key. In this case the following setup applies:

[[TPS\\_SWCT\\_01779](#)] `AtomicSwComponentType` uses the API of the Crypto Service to derive a key [

**ServiceNeeds kind** : `CryptoServiceJobNeeds`

**RoleBasedPortAssignment** valid roles:

- CsmJobKeyDerive\_{Primitive} [1]
- CallbackNotification [1]

#### RoleBasedDataAssignment

N/A

#### RepresentedPortGroups

N/A

]([RS\\_SWCT\\_00030](#))

Please note that the resolution of the name fragment {Primitive} that appears in the `shortName` of the `ClientServerInterface` `CsmJobKeyDerive_{Primitive}` is regulated by [[TPS\\_SWCT\\_01727](#)].

### 13.7.3.5 Crypto Service Use Case: usage of job API to execute calculation of the public value for key exchange

Scenario: a `AtomicSwComponentType` uses the **job API** of the Crypto Service to execute calculation of the public value for key exchange. In this case the following setup applies:

[TPS\_SWCT\_01780] [AtomicSwComponentType](#) uses the API of the Crypto Service to execute calculation of the public value for key exchange [

**ServiceNeeds kind** : [CryptoServiceJobNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [CsmJobKeyExchangeCalcPubVal\\_{Primitive}](#) [1]
- [CallbackNotification](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

Please note that the resolution of the name fragment {Primitive} that appears in the [shortName](#) of the [ClientServerInterface](#) [CsmJobKeyExchangeCalcPubVal\\_{Primitive}](#) is regulated by [TPS\_SWCT\_01727].

### 13.7.3.6 Crypto Service Use Case: usage of job API to execute calculation of shared secret for key exchange

Scenario: a [AtomicSwComponentType](#) uses the **job API** of the Crypto Service to execute calculation of shared secret for key exchange. In this case the following setup applies:

[TPS\_SWCT\_01781] [AtomicSwComponentType](#) uses the API of the Crypto Service to execute calculation of shared secret for key exchange [

**ServiceNeeds kind** : [CryptoServiceJobNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [CsmJobKeyExchangeCalcSecret\\_{Primitive}](#) [1]
- [CallbackNotification](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

Please note that the resolution of the name fragment {Primitive} that appears in the [shortName](#) of the [ClientServerInterface](#) [CsmJobKeyExchangeCalcSecret\\_{Primitive}](#) is regulated by [TPS\_SWCT\_01727].

### 13.7.3.7 Crypto Service Use Case: usage of job API to execute certificate parsing

Scenario: a [AtomicSwComponentType](#) uses the **job API** of the Crypto Service to execute certificate parsing. In this case the following setup applies:

**[TPS\_SWCT\_01782] [AtomicSwComponentType](#) uses the API of the Crypto Service to execute certificate parsing** [

**ServiceNeeds kind** : [CryptoServiceJobNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [CsmJobCertificateParse\\_{Primitive}](#) [1]
- [CallbackNotification](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

Please note that the resolution of the name fragment `{Primitive}` that appears in the `shortName` of the [ClientServerInterface](#) `CsmJobCertificateParse_{Primitive}` is regulated by [\[TPS\\_SWCT\\_01727\]](#).

### 13.7.3.8 Crypto Service Use Case: usage of job API to execute certificate verification

Scenario: a [AtomicSwComponentType](#) uses the **job API** of the Crypto Service to execute certificate verification. In this case the following setup applies:

**[TPS\_SWCT\_01783] [AtomicSwComponentType](#) uses the API of the Crypto Service to execute certificate verification** [

**ServiceNeeds kind** : [CryptoServiceJobNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [CsmJobCertificateVerify\\_{Primitive}](#) [1]
- [CallbackNotification](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

Please note that the resolution of the name fragment {Primitive} that appears in the `shortName` of the `ClientServerInterface CsmJobCertificateVerify_{Primitive}` is regulated by [TPS\_SWCT\_01727].

### 13.7.4 Crypto Key Management Use Cases

The service use cases for cryptographic key management are indicated by the presence of a `CryptoKeyManagementNeeds` aggregated at `SwcServiceDependency` in the role `serviceNeeds`.

<b>Class</b>	<b>CryptoKeyManagementNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class can be used to indicate a service use case for key management.			
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.10: CryptoKeyManagementNeeds**

#### 13.7.4.1 KeyM Service Use Case: Software-Component wants check a certificate on KeyM

Scenario: A software-component wants check a certificate on `KeyM`. The software-component sends the certificate to `KeyM (SetCertificate)` and then initiates a verification of the certificate (`VerifyCertificate`).

While the verification is still running it is possible to obtain the status of the verification (`GetStatus`).

[TPS\_SWCT\_01813] **Software-Component wants check a certificate on KeyM** [

**ServiceNeeds kind** : `CryptoKeyManagementNeeds`

**RoleBasedPortAssignment** valid roles:

- `KeyM_Certificate` [1]
- `KeyMCertificateNotification` [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](`RS_SWCT_00030`)

### 13.7.4.2 KeyM Service Use Case: Software-Component wants to retrieve a certificate from KeyM

Scenario: A software-component wants to retrieve a certificate on KeyM. The software-component request the certificate by using GetCertificate. This step represents a synchronous operation.

[TPS\_SWCT\_01814] **AtomicSwComponentType** wants to retrieve a certificate from KeyM [

**ServiceNeeds kind** : [CryptoKeyManagementNeeds](#)

**RoleBasedPortAssignment** valid roles:

- KeyM\_Certificate [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

### 13.7.4.3 KeyM Service Use Case: Software-Component wants to retrieve elements of a certificate from KeyM

Scenario: A software-component wants to retrieve elements of a certificate on KeyM (CertificateElementGet, CertificateElementGetByIndex, CertificateElementGetCount). This step represents a synchronous operation.

[TPS\_SWCT\_01815] **AtomicSwComponentType** wants to retrieve elements of a certificate from KeyM [

**ServiceNeeds kind** : [CryptoKeyManagementNeeds](#)

**RoleBasedPortAssignment** valid roles:

- KeyMCertificateElement [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))



#### 13.7.4.4 KeyM Service Use Case: Software-Component wants to check the existence of a certificate from KeyM

Scenario: A software-component wants to check the existence of a certificate from KeyM (GetStatus).

[TPS\_SWCT\_01816] **AtomicSwComponentType** wants to check the existence of a certificate from KeyM [

**ServiceNeeds kind** : [CryptoKeyManagementNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [KeyM\\_Certificate](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

#### 13.7.4.5 KeyM Service Use Case: Software-Component wants to store a (derived) key in KeyM

Scenario: A software-component wants to store a (derived) key in KeyM.

[TPS\_SWCT\_01817] **AtomicSwComponentType** wants to store a (derived) key in KeyM [

**ServiceNeeds kind** : [CryptoKeyManagementNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [KeyMCryptoKey](#) [1]
- [KeyMCryptoKeyNotification](#) [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

### 13.7.4.6 KeyM Service Use Case: Software-Component wants to store a container with (encrypted) keys in KeyM

Scenario: A software-component wants to store a container with (encrypted) keys in KeyM. The processing of the container is done in the context of a session that enables the usage of the `Start` and `Finish` operations.

By calling the `Prepare` operation the software-component passes the container to the KeyM, which reads the content of the container and extracts the contained cryptographic keys. The next step is that the KeyM updates every key extracted from the container.

**[TPS\_SWCT\_01818] AtomicSwComponentType wants to store a container with encrypted keys (e.g. She-keys) in KeyM** [

**ServiceNeeds kind** : [CryptoKeyManagementNeeds](#)

**RoleBasedPortAssignment** valid roles:

- `KeyMCryptoKey` [1]
- `KeyMCryptoKeyNotification` [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

### 13.7.4.7 KeyM Service Use Case: Software-Component wants to verify if cryptographic operation was executed using a specific key

Scenario: A software-component wants to verify if cryptographic operation was executed using a specific key (`Verify`).

**[TPS\_SWCT\_01819] AtomicSwComponentType wants to verify if cryptographic operation was executed using a specific key** [

**ServiceNeeds kind** : [CryptoKeyManagementNeeds](#)

**RoleBasedPortAssignment** valid roles:

- `KeyMCryptoKey` [1]
- `KeyMCryptoKeyNotification` [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

## 13.8 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 7.38 and can be divided into four main parts:

- Function Inhibition Needs in chapter 13.8.2
- Diagnostic Event Needs in chapter 13.8.3
- Diagnostic Communication Needs in chapter 13.8.4
- Service Needs to fulfill the OBD related requirements in chapter 13.8.5

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\_00170, RS\_SWCT\_03190*)

**[constr\_1640] No use of Optional Element Structure for interaction with the diagnostic stack** [An `SwcServiceDependency` that aggregates a diagnostic-related subclass of `ServiceNeeds` shall not refer to any `PortPrototype` by means of either a `RoleBasedPortAssignment` or `RoleBasedDataAssignment` where the respective `PortInterface` contains any `DataPrototype` typed by an `Optional Element Structure`.

This rule shall be imposed **at the time when the RTE is generated.**]()

### 13.8.1 Development Approach

AUTOSAR supports in many cases the implementation of a bottom-up or top-down configuration for various aspects of the development workflow.

Before the advent of the `Diagnostic Extract` [39], AUTOSAR officially supported a modeling approach where the configuration of an `ApplicationSwComponentType`

might contain contributions to the external configuration of the diagnostic stack. This was considered an approach to implement a top-down configuration approach for diagnostics.

However, the development approach introduced with the [Diagnostic Extract](#) turned out to be superior and less prone to mistakes and limitations in comparison to the configuration that each developer contributed without necessarily having the knowledge about the greater scope of diagnostic configuration.

Therefore, the usage of the [Diagnostic Extract](#) has become the canonical approach to the configuration of the external behavior of the AUTOSAR diagnostic stack and the respective configuration attributes available in the scope of `SwcServiceDependency` have been removed from the AUTOSAR methodology in order to reduce potential confusion in the audience.

In particular, a top-down approach using a [Diagnostic Extract](#) can be implemented if the [Diagnostic Extract](#) also provides the mappings between diagnostic elements and elements of the application model. If the mappings are not wanted or not available, it is also possible to use the [Diagnostic Extract](#) to derive a bottom-up configuration of the diagnostic stack without the relations to application software.

### 13.8.2 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 [FunctionInhibitionNeeds](#) as well as [FunctionInhibitionAvailabilityNeeds](#) elements in the context of an `SwcServiceDependency`. Each [FunctionInhibitionNeeds](#) and [FunctionInhibitionAvailabilityNeeds](#) defines the requirements related to one function inhibition ID (for the terms related to the AUTOSAR Function Inhibition Manager, see [40]).

<b>Class</b>	<b>FunctionInhibitionNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.11: FunctionInhibitionNeeds**

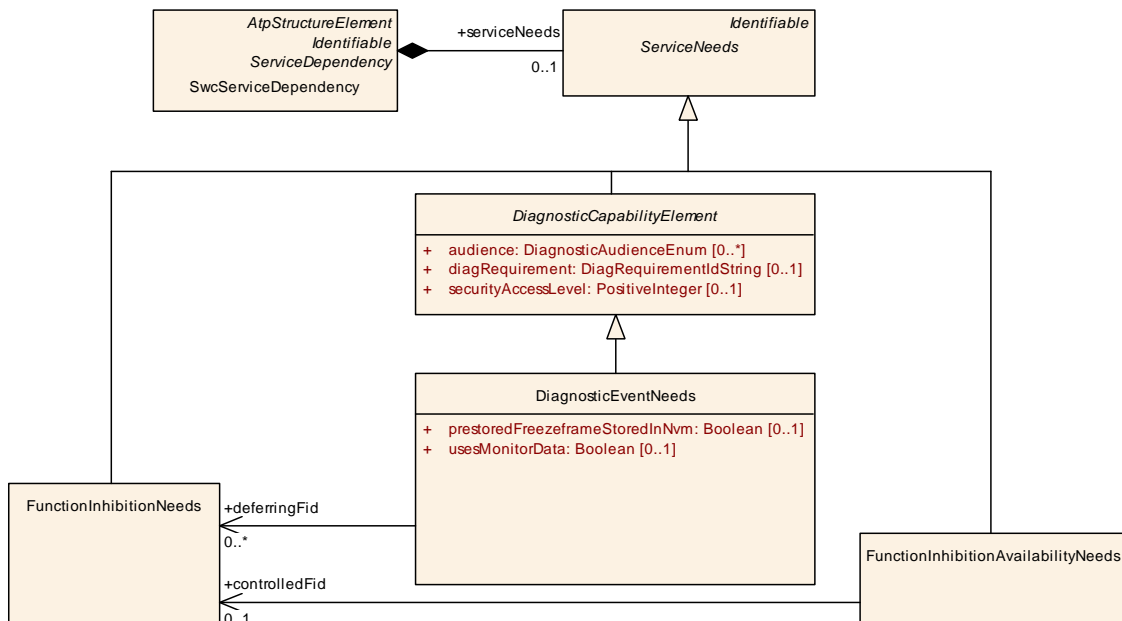
<b>Class</b>	<b>FunctionInhibitionAvailabilityNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the abstract needs on the configuration of the Function Inhibition Manager to provide the control function for one Function Identifier (FID).			





<b>Class</b>	<b>FunctionInhibitionAvailabilityNeeds</b>			
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
controlledFid	<a href="#">FunctionInhibitionNeeds</a>	0..1	ref	This reference represents the controlled FID

**Table 13.12: FunctionInhibitionAvailabilityNeeds**



**Figure 13.5: Modeling of [FunctionInhibitionNeeds](#) and [FunctionInhibitionAvailabilityNeeds](#)**

### 13.8.2.1 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

|(RS\_SWCT\_00170, RS\_SWCT\_03190)

For more information please refer to [SWS\_Fim\_00090].

### 13.8.2.2 Function Inhibition Manager Use Case: react on suppressed or unavailable events

**[TPS\_SWCT\_01739] Function Inhibition Manager Use Case: react on suppressed or unavailable events** [Scenario: an `AtomicSwComponentType` wants to react on suppressed or unavailable events and disable the permission to run for a FID. In this case, the following setup applies:

**ServiceNeeds kind** `FunctionInhibitionAvailabilityNeeds`

**RoleBasedPortAssignment** valid roles:

- `ControlFunctionAvailable` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

|(RS\_SWCT\_00170, RS\_SWCT\_03190)

Note: for variant coding `ClientServerInterface` `ControlFunctionAvailable` is used to deactivate a certain functionality (e.g. to set the FID to not available).

For more information, please refer to [SWS\_Fim\_00107].

### 13.8.3 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 [41]).

<b>Class</b>	<code>DiagnosticEventManagerNeeds</code>
<b>Package</b>	<code>M2::AUTOSARTemplates::CommonStructure::ServiceNeeds</code>
<b>Note</b>	Specifies the general needs on the configuration of the Diagnostic Event Manager (Dem) which are not related to a particular item.
<b>Base</b>	<code>ARObject</code> , <code>DiagnosticCapabilityElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code> , <code>ServiceNeeds</code>



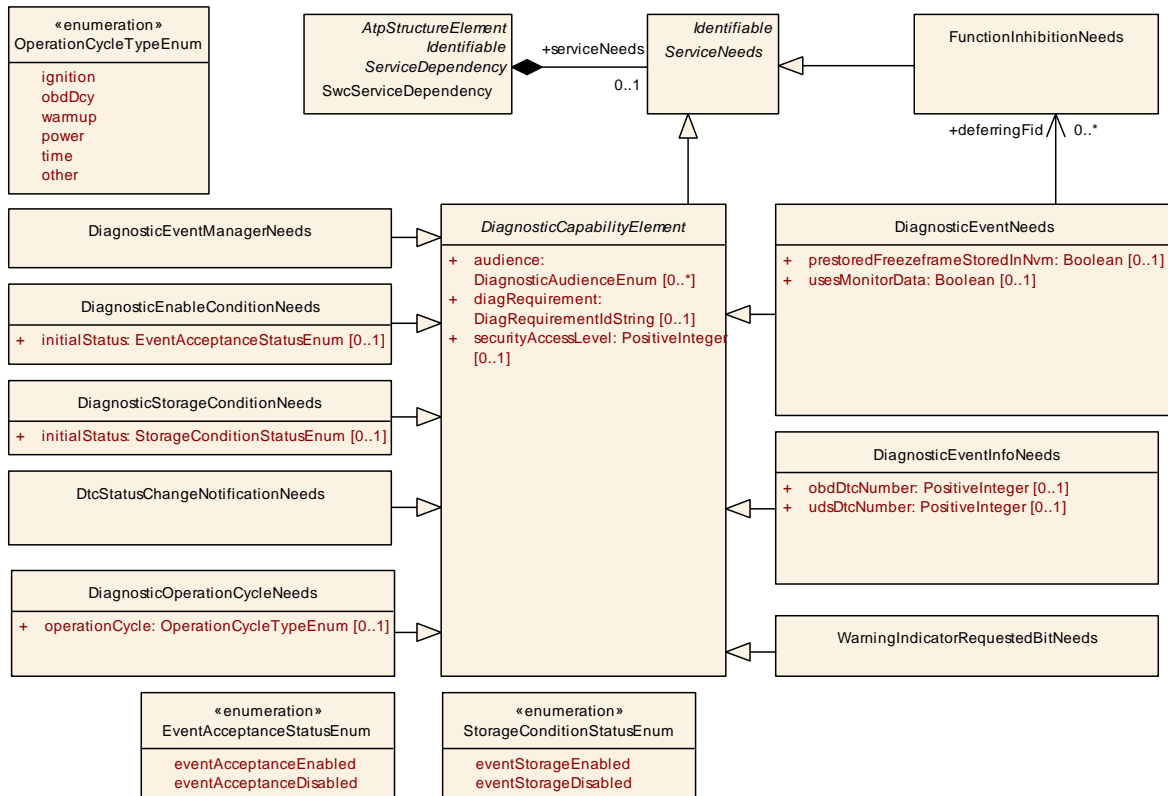


Class	DiagnosticEventManagerNeeds			
Attribute	Type	Mult.	Kind	Note
-	-	-	-	-

**Table 13.13: 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.



**Figure 13.6: General diagnostic event-handling related [ServiceNeeds](#)**

<b>Class</b>	<i>DiagnosticCapabilityElement</i> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	This class identifies the capability to provide generic information about diagnostic capabilities
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>





<b>Class</b>	<b>DiagnosticCapabilityElement</b> (abstract)			
<b>Subclasses</b>	<a href="#">DiagnosticCommunicationManagerNeeds</a> , <a href="#">DiagnosticComponentNeeds</a> , <a href="#">DiagnosticControlNeeds</a> , <a href="#">DiagnosticEnableConditionNeeds</a> , <a href="#">DiagnosticEventInfoNeeds</a> , <a href="#">DiagnosticEventManagerNeeds</a> , <a href="#">DiagnosticEventNeeds</a> , <a href="#">DiagnosticIoControlNeeds</a> , <a href="#">DiagnosticOperationCycleNeeds</a> , <a href="#">DiagnosticRequestFileTransferNeeds</a> , <a href="#">DiagnosticResponseOnEventNeeds</a> , <a href="#">DiagnosticRoutineNeeds</a> , <a href="#">DiagnosticStorageConditionNeeds</a> , <a href="#">DiagnosticUploadDownloadNeeds</a> , <a href="#">DiagnosticValueNeeds</a> , <a href="#">DiagnosticsCommunicationSecurityNeeds</a> , <a href="#">DtcStatusChangeNotificationNeeds</a> , <a href="#">ObdControlServiceNeeds</a> , <a href="#">ObdInfoServiceNeeds</a> , <a href="#">ObdMonitorServiceNeeds</a> , <a href="#">ObdPidServiceNeeds</a> , <a href="#">ObdRatioDenominatorNeeds</a> , <a href="#">ObdRatioServiceNeeds</a> , <a href="#">WarningIndicatorRequestedBitNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
audience	<a href="#">DiagnosticAudienceEnum</a>	*	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.
diag Requirement	DiagRequirementId String	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.
securityAccess Level	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 13.14: DiagnosticCapabilityElement**

<b>Enumeration</b>	<b>DiagnosticAudienceEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	The possible values of the intended audience for a diagnostic object.
<b>Literal</b>	<b>Description</b>
aftermarket	The object is for free aftermarket service organizations. <b>Tags:</b> atp.EnumerationLiteralIndex=1
afterSales	The object is relevant for the OEM after-sales organization. <b>Tags:</b> atp.EnumerationLiteralIndex=2
development	The object is relevant for engineering only. <b>Tags:</b> atp.EnumerationLiteralIndex=3
manufacturing	The object is relevant for manufacturing. <b>Tags:</b> atp.EnumerationLiteralIndex=4
supplier	The object is relevant for the ECU-supplier aftermarket organization. <b>Tags:</b> atp.EnumerationLiteralIndex=5

**Table 13.15: 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 [41]).

In addition, [ObdPidServiceNeeds](#) and [ObdRatioServiceNeeds](#) are required in order to specify the needs for OBD diagnostic service calls.



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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

**[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 `CallbackGetFault-DetectCounter`.

This rule shall be imposed **at the time when the RTE is generated.**]()

<b>Class</b>	<b>DiagnosticEventNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the abstract needs on the configuration of the Diagnostic Event Manager for one diagnostic event. Its shortName 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 shortName shall be the name of the production error.			
<b>Base</b>	<i>ARObject, DiagnosticCapabilityElement, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
deferringFid	<a href="#">FunctionInhibitionNeeds</a>	*	ref	This reference contains the link to a function identifier within the FiM which is used by the monitor before delivering a result.
diagEventDebounceAlgorithm	<a href="#">DiagEventDebounceAlgorithm</a>	0..1	aggr	Specifies the abstract need on the Debounce Algorithm applied by the Diagnostic Event Manager.
prestoredFreezeFrameStoredInNvm	Boolean	0..1	attr	If the Event uses a prestored freeze-frame (using the operations <code>PrestoreFreezeFrame</code> and <code>ClearPrestoredFreezeFrame</code> of the service interface <code>DiagnosticMonitor</code> ) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm).
usesMonitorData	Boolean	0..1	attr	This attribute defines whether additional monitor data shall be added to the reporting of events.

**Table 13.16: DiagnosticEventNeeds**

<b>Class</b>	<i>DiagEventDebounceAlgorithm</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	<p>This class represents the ability to specify the pre-debounce algorithm which is selected and/or required by the particular monitor.</p> <p>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.</p>			
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Subclasses</b>	<i>DiagEventDebounceCounterBased</i> , <i>DiagEventDebounceMonitorInternal</i> , <i>DiagEventDebounceTimeBased</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.17: DiagEventDebounceAlgorithm**

<b>Class</b>	<b>DiagEventDebounceCounterBased</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	<p>This meta-class represents the ability to indicate that the counter-based debounce algorithm shall be used by the DEM for this diagnostic monitor.</p> <p>This is related to set the ECUC choice container DemDebounceAlgorithmClass to DemDebounceCounterBased.</p>			
<b>Base</b>	<i>ARObject</i> , <i>DiagEventDebounceAlgorithm</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
counterBasedFdcThresholdStorageValue	Integer	0..1	attr	Threshold to allocate an event memory entry and to capture the Freeze Frame.
counterDecrementStepSize	Integer	0..1	attr	<p>This value shall be taken to decrement the internal debounce counter.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
counterFailedThreshold	Integer	0..1	attr	<p>This value defines the event-specific limit that indicates the "failed" counter status.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
counterIncrementStepSize	Integer	0..1	attr	<p>This value shall be taken to increment the internal debounce counter.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
counterJumpDown	Boolean	0..1	attr	<p>This value activates or deactivates the counter jump-down behavior.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
counterJumpDownValue	Integer	0..1	attr	<p>This value represents the initial value of the internal debounce counter if the counting direction changes from incrementing to decrementing.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
counterJumpUp	Boolean	0..1	attr	<p>This value activates or deactivates the counter jump-up behavior.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>





Class	DiagEventDebounceCounterBased			
counterJumpUp Value	Integer	0..1	attr	This value represents the initial value of the internal debounce counter if the counting direction changes from decrementing to incrementing. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
counterPassed Threshold	Integer	0..1	attr	This value defines the event-specific limit that indicates the "passed" counter status. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

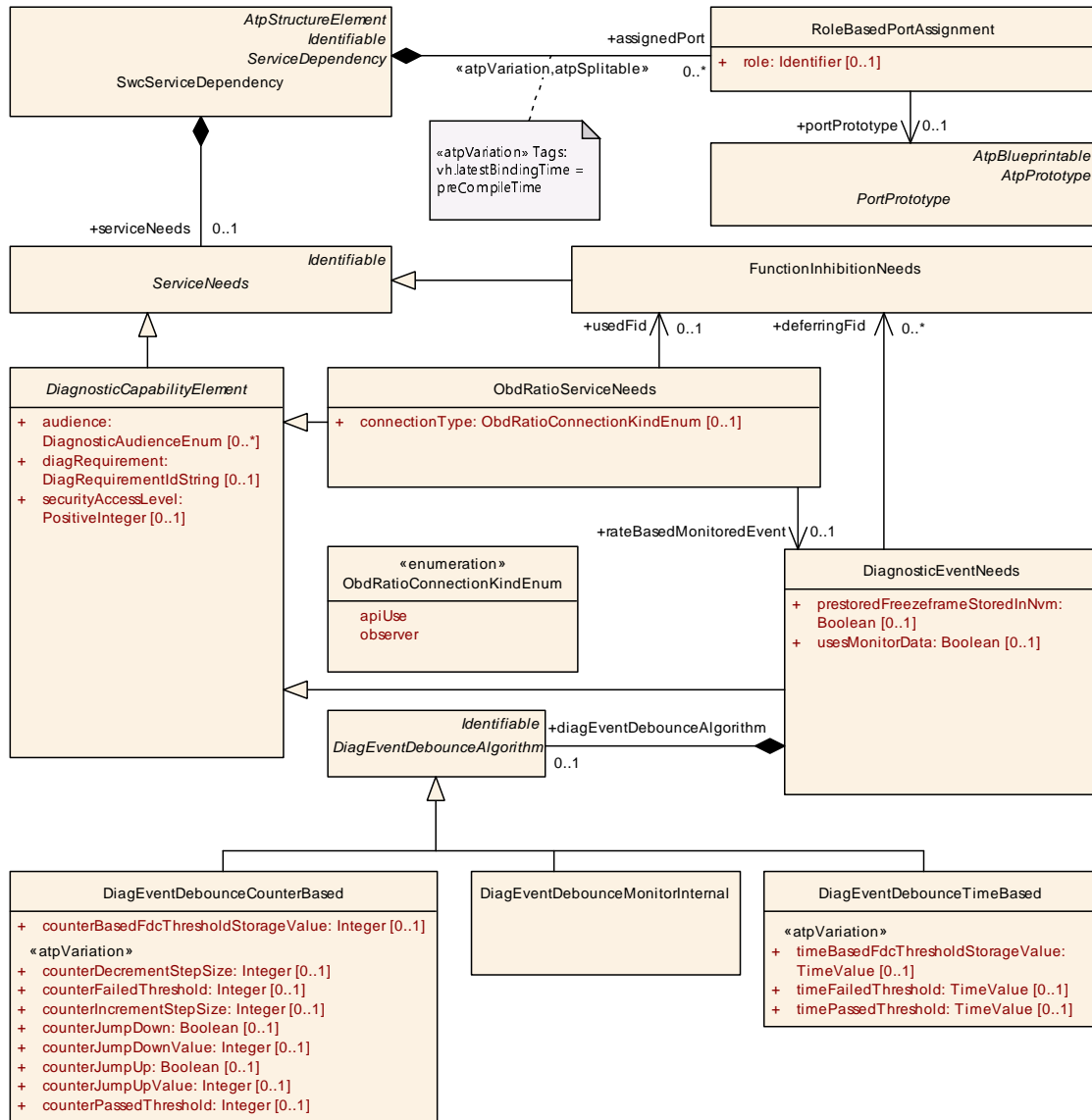
**Table 13.18: DiagEventDebounceCounterBased**

Class	DiagEventDebounceTimeBased			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">DiagEventDebounceAlgorithm</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
timeBasedFdc Threshold StorageValue	TimeValue	0..1	attr	Threshold to allocate an event memory entry and to capture the Freeze Frame. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
timeFailed Threshold	TimeValue	0..1	attr	This value represents the event-specific delay indicating the "failed" status. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
timePassed Threshold	TimeValue	0..1	attr	This value represents the event-specific delay indicating the "passed" status. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 13.19: DiagEventDebounceTimeBased**

Class	DiagEventDebounceMonitorInternal			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	"This meta-class represents the ability to indicate that no Dem pre-debounce algorithm shall be used for this diagnostic monitor. The SWC might implement an internal debouncing algorithm and report qualified (debounced) results to the Dem/DM.			
<b>Base</b>	ARObject, <a href="#">DiagEventDebounceAlgorithm</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

**Table 13.20: DiagEventDebounceMonitorInternal**



**Figure 13.7: Relationship of DiagnosticEventNeeds and FunctionInhibitionNeeds**

The figure 13.7 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:

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 13.8.5

**[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 [40]), 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.

<b>Class</b>	<b>DiagnosticEventInfoNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the needs of a software-component interested to get information regarding specific DTCs.			
<b>Base</b>	<i>ARObject</i> , <i>DiagnosticCapabilityElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>Service Needs</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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 13.21: DiagnosticEventInfoNeeds**

<b>Class</b>	<b>DiagnosticOperationCycleNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the needs of a software-component to provide information regarding the operation cycle management to the <code>Dem</code> module.			
<b>Base</b>	<i>ARObject</i> , <i>DiagnosticCapabilityElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>Service Needs</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
operationCycle	OperationCycleTypeEnum	0..1	attr	Operation cycles types for the <code>Dem</code> to be supported by cycle-state APIs.

**Table 13.22: DiagnosticOperationCycleNeeds**

<b>Enumeration</b>	<b>OperationCycleTypeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	The possible values of the operation cycles types for the Dem.
<b>Literal</b>	<b>Description</b>
ignition	Ignition ON / OFF cycle. <b>Tags:</b> atp.EnumerationLiteralIndex=0
obdDcy	OBd Driving cycle. <b>Tags:</b> atp.EnumerationLiteralIndex=1
other	Further operation cycle. <b>Tags:</b> atp.EnumerationLiteralIndex=2
power	Power ON / OFF cycle. <b>Tags:</b> atp.EnumerationLiteralIndex=3
time	Time based operation cycle. <b>Tags:</b> atp.EnumerationLiteralIndex=4
warmup	OBd Warm up cycle. <b>Tags:</b> atp.EnumerationLiteralIndex=5

**Table 13.23: OperationCycleTypeEnum**

<b>Class</b>	<b>DiagnosticEnableConditionNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the needs of a software-component to provide the capability to set an enable condition.			
<b>Base</b>	<i>ARObject</i> , <i>DiagnosticCapabilityElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>Service Needs</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initialStatus	<a href="#">EventAcceptanceStatus Enum</a>	0..1	attr	Defines the initial status for enable or disable of acceptance of event reports of a diagnostic event.

**Table 13.24: DiagnosticEnableConditionNeeds**

<b>Enumeration</b>	<b>EventAcceptanceStatusEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	This enumerator specifies the initial status for enable or disable of acceptance of event reports of a diagnostic event.
<b>Literal</b>	<b>Description</b>
eventAcceptance Disabled	Acceptance of a diagnostic event is disabled. <b>Tags:</b> atp.EnumerationLiteralIndex=0
eventAcceptance Enabled	Acceptance of a diagnostic event is enabled. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 13.25: EventAcceptanceStatusEnum**

<b>Class</b>	<b>DiagnosticStorageConditionNeeds</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds





<b>Class</b>	<b>DiagnosticStorageConditionNeeds</b>			
<b>Note</b>	This meta-class represents the needs of a software-component to provide the capability to set a storage condition.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initialStatus	<a href="#">StorageConditionStatusEnum</a>	0..1	attr	Defines the initial status for enable or disable of storage of a diagnostic event.

**Table 13.26: DiagnosticStorageConditionNeeds**

<b>Enumeration</b>	<b>StorageConditionStatusEnum</b>		
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds		
<b>Note</b>	This enumeration specifies the initial status for enable or disable of storage of a diagnostic event.		
<b>Literal</b>	<b>Description</b>		
eventStorage Disabled	Storage of a diagnostic event is disabled. <b>Tags:</b> atp.EnumerationLiteralIndex=0		
eventStorage Enabled	Storage of a diagnostic event is enabled. <b>Tags:</b> atp.EnumerationLiteralIndex=1		

**Table 13.27: StorageConditionStatusEnum**

<b>Class</b>	<b>DtcStatusChangeNotificationNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the needs of a software-component interested to get information regarding any DTC status change.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.28: DtcStatusChangeNotificationNeeds**

### 13.8.3.1 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]
- `CallbackEventUdsStatusChanged` [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`.

**13.8.3.2 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]
- `CallbackEventUdsStatusChanged` [0..1]
- `CallbackClearEventAllowed` [0..1]
- `CallbackGetFaultDetectCounter` [0..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`.



### 13.8.3.3 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].

### 13.8.3.4 Dem Service Use Case: software-component enables reporting of DTCs in general

Scenario: a [AtomicSwComponentType](#) enables the reporting of DTCs in general.

[TPS\_SWCT\_01134] [AtomicSwComponentType](#) enables reporting 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].

**13.8.3.5 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

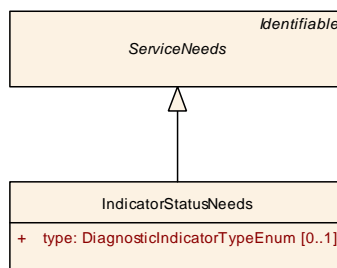
](`RS_SWCT_00170`, `RS_SWCT_03190`)

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.

**13.8.3.6 Dem Service Use Case: retrieve information of the lamp status**

Scenario: an `AtomicSwComponentType` retrieves information of the lamp status.



**Figure 13.8: Modeling of `IndicatorStatusNeeds`**

[TPS\_SWCT\_01136] `AtomicSwComponentType` retrieves information of the lamp status [

**ServiceNeedsKind** `IndicatorStatusNeeds`

**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].

<b>Class</b>	<b>IndicatorStatusNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class shall be taken to signal a service use case that affects the indicator status.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
type	<a href="#">DiagnosticIndicatorType Enum</a>	0..1	attr	Defines the type of the indicator.

**Table 13.29: IndicatorStatusNeeds**

<b>Enumeration</b>	<b>DiagnosticIndicatorTypeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticIndicator
<b>Note</b>	Type of an indicator.
<b>Literal</b>	<b>Description</b>
amberWarning	Amber Warning Lamp <b>Tags:</b> atp.EnumerationLiteralIndex=0
malfunction	Malfunction Indicator Lamp <b>Tags:</b> atp.EnumerationLiteralIndex=1
protectLamp	Protect Lamp <b>Tags:</b> atp.EnumerationLiteralIndex=2
redStopLamp	Red Stop Lamp <b>Tags:</b> atp.EnumerationLiteralIndex=3
warning	Warning <b>Tags:</b> atp.EnumerationLiteralIndex=4

**Table 13.30: DiagnosticIndicatorTypeEnum**

### 13.8.3.7 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].

**13.8.3.8 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].

**13.8.3.9 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].

**13.8.3.10 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, it is advised to create a [SwcServiceDependency](#) for each kind of status change.

**13.8.3.11 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].

### 13.8.3.12 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 [13.8.3.11](#) or [13.8.3.10](#) 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]
- [GeneralCallbackEventUdsStatusChange](#) [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](#))

### 13.8.3.13 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 [41] 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].

### 13.8.3.14 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]`.

### 13.8.3.15 Dem Service Use Case: Software-Component wants to be triggered on Monitor Status Changes

Scenario: a software-component wants to be triggered on monitor status changes if this is supported for the specific monitor status. Events reported from basic-software modules cannot be considered in this service use case.

The Dem will not provide corresponding `PortPrototypes` for events reported by basic-software modules.

This way, the service use case cannot be used for events reported by the basic-software.

However, for the creator of the service use case there is no way to find out whether the event will be reported by basic software of application software-component.

#### **[TPS\_SWCT\_01715] Software-Component wants to be triggered on Monitor Status Changes** [

**ServiceNeeds** kind `DiagnosticEventManagerNeeds`

**RoleBasedPortAssignment** valid roles:

- `CallbackMonitorStatusChange` [1]



**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

**13.8.3.16 Dem Service Use Case: write parameter identifier by software-component**

Scenario: A software-component computes the PIDs, and pushes them to Dem for storage and reporting to Dcm.

**[TPS\_SWCT\_01766] Software-component computes the PIDs, and pushes them to Dem for storage and reporting to Dcm** [

**ServiceNeeds** kind [ObdPidServiceNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [SetDataOfPID21](#) [1]
- [SetDataOfPID4D](#) [1]
- [SetDataOfPID4E](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

**13.8.3.17 Dem Service Use Case: read parameter identifier by software-component**

Scenario: A software-component located on an `OBD master ECU` reads the PID 21, and sends the value around via “regular” sender-receiver communication to other software-components located on `OBD primary ECUs` with the obligation to push the PID value to their local Dem.

**[TPS\_SWCT\_01767] Software-component located on an OBD master ECU reads the PID 21, and sends the value around via “regular” sender-receiver communication** [

**ServiceNeeds** kind [ObdPidServiceNeeds](#)

**RoleBasedPortAssignment** valid roles:

- GetDataOfPID21 [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

|(RS\_SWCT\_00170, RS\_SWCT\_03190)

**13.8.3.18 Dem Service Use Case: diagnostic monitor provides monitor data, debouncing by Dem**

Scenario: an [AtomicSwComponentType](#) implements a Diagnostic Monitor that is able to provide monitor data. The debouncing of the failure condition shall be configured and processed by the Dem. In this case the following setup applies:

**[TPS\_SWCT\_01789] [AtomicSwComponentType](#) implements a Diagnostic Monitor that provides monitor data, debouncing by Dem** [

**ServiceNeeds** kind [DiagnosticEventNeeds](#) (with attribute [usesMonitorData](#) set to TRUE)

**RoleBasedPortAssignment** valid roles:

- DiagnosticMonitor [0..1]
- DiagnosticMonitor\_MonitorData [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 the sub-class [DiagEventDebounceCounterBased](#) or [DiagEventDebounceTimeBased](#) algorithm should be used as the value for attribute [DiagnosticEventNeeds.diagEventDebounceAlgorithm](#).

### 13.8.3.19 Dem Service Use Case: diagnostic monitor provides monitor data, debouncing by software-component

Scenario: an `AtomicSwComponentType` implements a Diagnostic Monitor that is able to provide monitor data. The debouncing of the failure condition shall be configured and processed by the software-component that implements the monitor. In this case the following setup applies:

**[TPS\_SWCT\_01790] `AtomicSwComponentType` implements a Diagnostic Monitor that provides monitor data, debouncing by software-component [**

**ServiceNeeds kind `DiagnosticEventNeeds` (with attribute `usesMonitorData` set to `TRUE`)**

**RoleBasedPortAssignment** valid roles:

- `DiagnosticMonitor` [0..1]
- `DiagnosticMonitor_MonitorData` [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 the sub-class `DiagEventDebounceMonitorInternal` should be used as the value for attribute `DiagnosticEventNeeds.diagEventDebounceAlgorithm`.

### 13.8.3.20 Dem Service Use Case: software-component checks whether an event is suppressed

Scenario: a software-component representing a monitor of a given event needs to check for the availability of the event in order to decide whether reporting of that event is cleared by the Dem.

A typical use case for this ability is a scenario where actuators are moved to their limits for testing purposes. During this phase the monitoring shall cease.

For this purpose the software component needs to expose an `RPortPrototype` towards the Dem.

**[TPS\_SWCT\_01808] Dem Service Use Case: software-component checks whether an event is suppressed [**

**ServiceNeeds** kind `DiagnosticEventInfoNeeds`

**RoleBasedPortAssignment** valid roles:

- `DiagnosticInfo` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

### 13.8.4 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 [42]).

The meta-class `DiagnosticRoutineNeeds` is used to define requirements to configure the Diagnostic Communication Manager Service. A `PPortPrototype` typed by a `ClientServerInterface`<sup>1</sup> may provide `ClientServerOperations` (for example, “start”, “stop”, and “RequestResults”).

The `PPortPrototype` corresponds to the diagnostic service `RoutineControl`. Within the `SwcInternalBehavior` up to three `RunnableEntitys` 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.

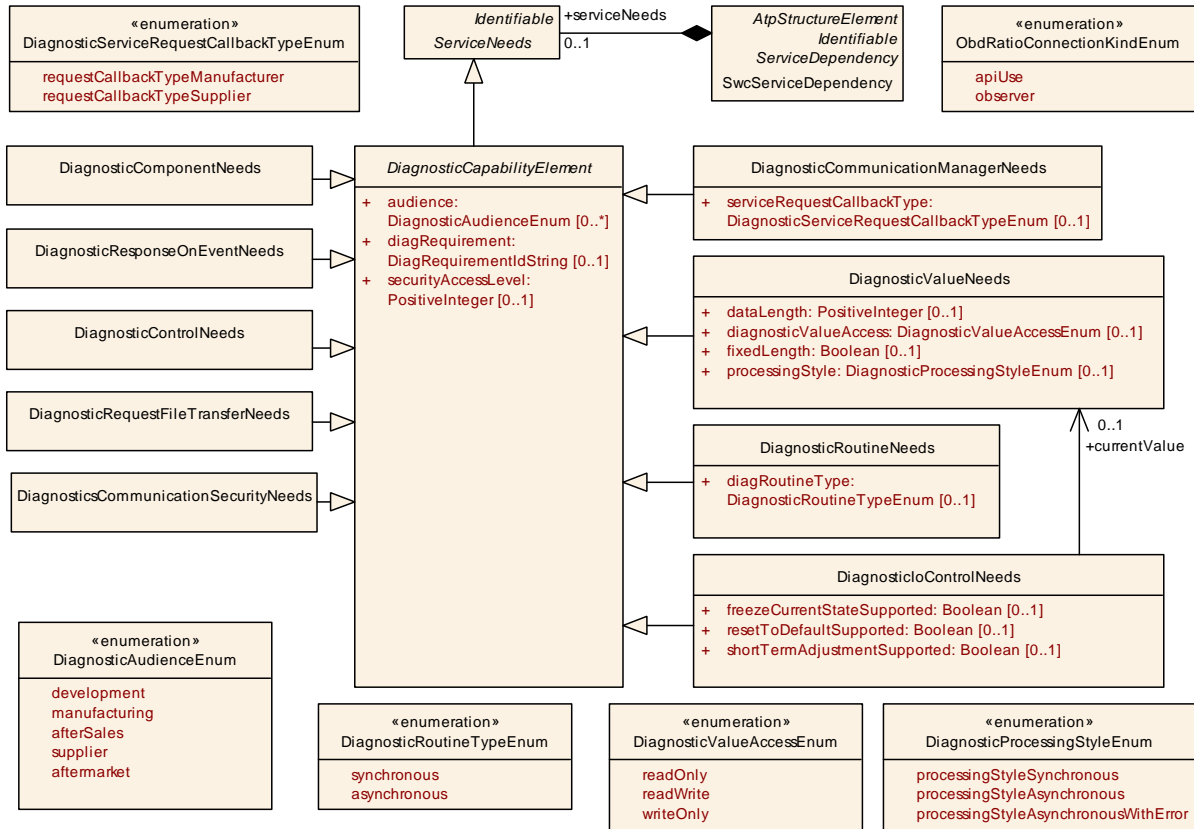
Please note that [constr\_1340] and [constr\_1341] apply for the application of `DiagnosticRoutineNeeds`. These constraints are part of the specification of the `Diagnostic Extract`.

---

<sup>1</sup>where `isService` shall be set to true

<b>Class</b>	<b>DiagnosticCommunicationManagerNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	ARObject, DiagnosticCapabilityElement, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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 13.31: DiagnosticCommunicationManagerNeeds**



**Figure 13.9: General diagnostic service-related ServiceNeeds**

<b>Enumeration</b>	<b>DiagnosticServiceRequestCallbackTypeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	This represents the ability to define whether a Service Request Notification was used in the role of a manufacturer or a supplier.
<b>Literal</b>	<b>Description</b>
requestCallbackTypeManufacturer	This represents the case that the usage of PortInterface ServiceRequestNotification has the characteristics of being used by a manufacturer. <b>Tags:</b> atp.EnumerationLiteralIndex=0





<b>Enumeration</b>	<b>DiagnosticServiceRequestCallbackTypeEnum</b>
requestCallbackTypeSupplier	This represents the case that the usage of PortInterface ServiceRequestNotification has the characteristics of being used by a supplier. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 13.32: DiagnosticServiceRequestCallbackTypeEnum**

<b>Class</b>	<b>DiagnosticRoutineNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	ARObject, DiagnosticCapabilityElement, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagRoutineType	DiagnosticRoutineTypeEnum	0..1	attr	This denotes the type of diagnostic routine which is implemented by the referenced server port.

**Table 13.33: DiagnosticRoutineNeeds**

**[constr\_1986] Existence of the reference DiagnosticRoutineNeeds.diagRoutineType** [For each DiagnosticRoutineNeeds, the reference to BswInternalBehavior in the role diagRoutineType shall exist **at the time when the RTE is generated.**] ()

<b>Enumeration</b>	<b>DiagnosticRoutineTypeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	This enumerator specifies the different types of diagnostic routines.
<b>Literal</b>	<b>Description</b>
asynchronous	This indicates that the diagnostic server is not blocked while the diagnostic routine is running. <b>Tags:</b> atp.EnumerationLiteralIndex=0
synchronous	This indicates that the diagnostic routine blocks the diagnostic server in the ECU while the routine is running. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 13.34: 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.

<b>Class</b>	<b>DiagnosticIoControlNeeds</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds





Class	DiagnosticIoControlNeeds			
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, MultilanguageReferrable, Referrable, ServiceNeeds			
Attribute	Type	Mult.	Kind	Note
currentValue	DiagnosticValueNeeds	0..1	ref	Reference to the DiagnosticValueNeeds indicating the access to the current value via signalBasedDiagnostics.
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 13.35: 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 **at any time in the workflow.**]()

**[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 **at any time in the workflow.**]()

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](#).

<b>Class</b>	<b>DiagnosticValueNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	<p>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.</p> <p>In the case of using a sender receiver communicated value, the related value shall be taken via assigned Data in the role "signalBasedDiagnostics".</p> <p>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).</p>			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">Service Needs</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataLength	PositiveInteger	0..1	attr	<p>This attribute is applicable only if the DiagnosticValue Needs is aggregated within a BswModuleDependency.</p> <p>This attribute represents the length of data (in bytes) provided for this particular PID signal.</p>
diagnosticValue Access	<a href="#">DiagnosticValueAccess Enum</a>	0..1	attr	<p>This attribute is applicable only if the DiagnosticValue Needs is aggregated within a BswModuleDependency.</p> <p>This attribute controls whether the data can be read and written or whether it is to be handled read-only.</p>
fixedLength	Boolean	0..1	attr	<p>This attribute is applicable only if the DiagnosticValue Needs is aggregated within a BswModuleDependency.</p> <p>This attribute controls whether the data length of the data is fixed.</p>
processingStyle	<a href="#">DiagnosticProcessing StyleEnum</a>	0..1	attr	<p>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.</p>

**Table 13.36: DiagnosticValueNeeds**

<b>Enumeration</b>	<b>DiagnosticValueAccessEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	Defines the access of the configured diagnostic current values which will be used by the Dem or Dcm module.
<b>Literal</b>	<b>Description</b>
readOnly	<p>The access to the data element is limited to read-only. This is typically used to read-out diagnostic information (e.g. current values).</p> <p><b>Tags:</b>atp.EnumerationLiteralIndex=0</p>
readWrite	<p>The value of the diagnostic data element is classified as configurable (read and write access is possible).</p> <p><b>Tags:</b>atp.EnumerationLiteralIndex=1</p>
writeOnly	<p>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,</p> <p><b>Tags:</b>atp.EnumerationLiteralIndex=2</p>

**Table 13.37: DiagnosticValueAccessEnum**



<b>Enumeration</b>	<b>DiagnosticProcessingStyleEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	This meta-class represents the ability to define the processing style of diagnostic requests.
<b>Literal</b>	<b>Description</b>
processingStyle Asynchronous	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. <b>Tags:</b> atp.EnumerationLiteralIndex=0
processingStyle AsynchronousWith Error	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 or handle error code. <b>Tags:</b> atp.EnumerationLiteralIndex=1
processingStyle Synchronous	The software-component is supposed to react synchronously on the request. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 13.38: DiagnosticProcessingStyleEnum**

<b>Class</b>	<b>DiagnosticsCommunicationSecurityNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the needs of a software-component to verify the access to security level via diagnostic services.			
<b>Base</b>	<i>AObject</i> , <i>DiagnosticCapabilityElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.39: DiagnosticsCommunicationSecurityNeeds**

### 13.8.4.1 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].

#### 13.8.4.2 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].

#### 13.8.4.3 Dcm Service Use Case: read/write current values by Sender Receiver Interface or Nv Data Interface

Scenario: an `AtomicSwComponentType` offers `PortPrototypes` typed by `SenderReceiverInterfaces` or `NvDataInterfaces` 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` or `NvDataInterfaces` 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 `[TPS_SWCT_01579]`, `[TPS_SWCT_01831]` and `[SWS_Dcm_00687]`.

**[constr\_1679] Existence of attribute `RoleBasedDataAssignment.usedDataElement.localVariable` for `RoleBasedDataAssignment.role = signalBasedDiagnostics` [If the attribute `RoleBasedDataAssignment.role` is set to the value `signalBasedDiagnostics` then the reference `RoleBasedDataAssignment.usedDataElement.localVariable` shall not exist at the time when the RTE is generated.]()**

For explanation of the existence of `[constr_1679]`, it is not intended to provide diagnostic access to local variables inside the `SwcInternalBehavior`.

#### 13.8.4.4 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](#)].

**[constr\_1724] Usage of attribute [ClientServerOperation.diagArgIntegrity](#)** [With the exception of the context of a [ServiceSwComponentType](#), the attribute [ClientServerOperation.diagArgIntegrity](#) shall only have the value `True` if the [ClientServerInterface](#) containing the respective [ClientServerOperation](#) is used to type a [PPortPrototype](#) that is referenced by a [RoleBasedPortAssignment](#) aggregated by a [SwcServiceDependency](#) that in turn aggregates [DiagnosticRoutineNeeds](#).

This rule shall be imposed **at the time when the RTE is generated.**]()

In other words, on the level of application software the attribute [diagArgIntegrity](#) shall only exist in the context of a [PPortPrototype](#).

Obviously, the [ServiceSwComponentType](#) that is used to represent the Dcm will use an [RPortPrototype](#) for the other end of the [AssemblySwConnector](#) that connects to said [PortPrototype](#) in the application software.

#### 13.8.4.5 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].

### 13.8.4.6 Dcm Service Use Case: IO control by Sender Receiver Interface

This use case represents an alternative to the use case described in chapter 13.8.4.5, 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 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]
- `DataServices` [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 `DiagnosticIoControlNeeds` 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](#))

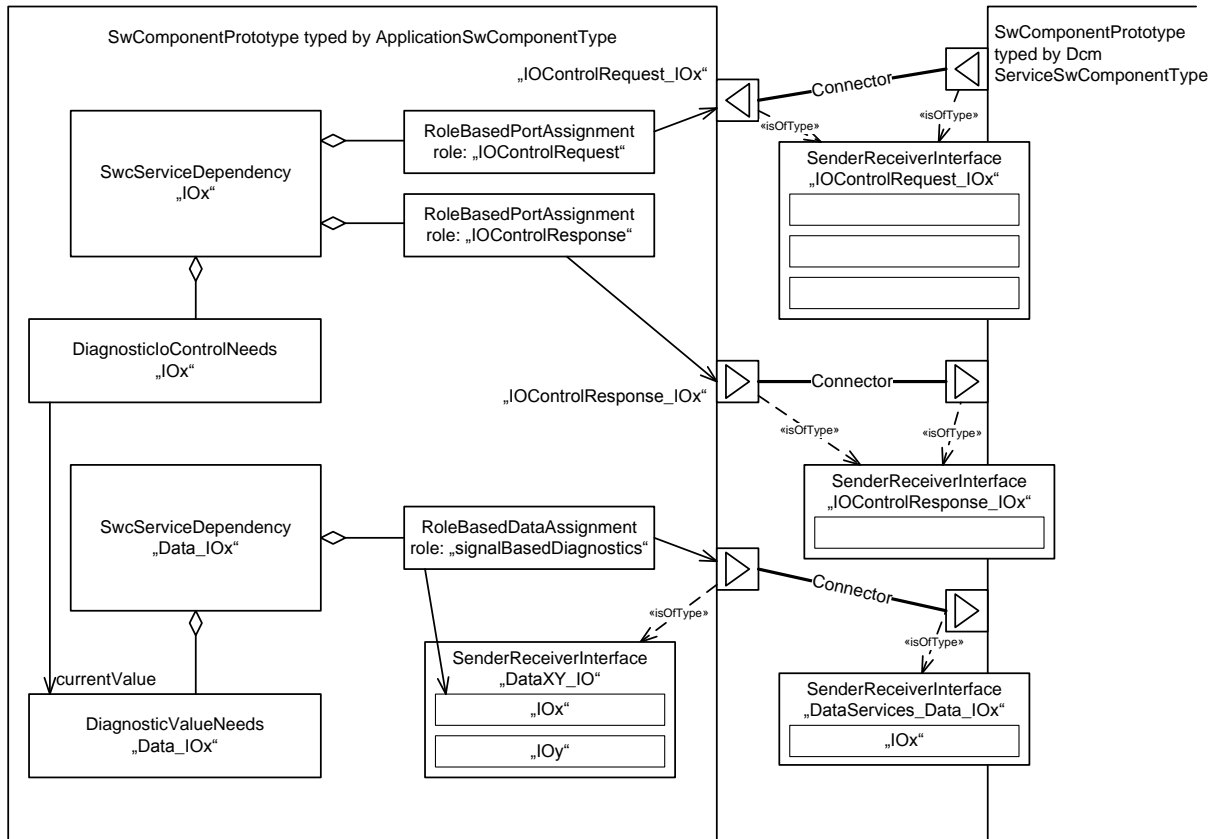
**[TPS\_SWCT\_01656] Suffix used for the resulting name of the `PortInterface` for `DataServices`, `IOControlRequest`, and `IOControlResponse`** [The *suffix* used for the resulting name of the `PortInterface` for the `DataServices_{Data}`, `IOControlRequest_{Data}`, and `IOControlResponse_{Data}` shall be taken from the `shortName` of the applicable `SwcServiceDependency`.] ([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

The service use case is visualized in Figure 13.10. The `SwComponentPrototype` contains two `SwcServiceDependencies`, one for the I/O Control, and one for the access of the `dataElement` with the `shortName` “IOx” by the Dcm.

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.



**Figure 13.10: Visualization of the service use case**

[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**)

For more information please refer to [SWS\_Dcm\_01308] and [SWS\_Dcm\_01309].

[constr\_1741] Restriction to explicit sending semantics for the usage of **DataServices** in the context of a **SwcServiceDependency** that aggregates **DiagnosticValueNeeds** that in turn is referenced by a **DiagnosticIoControlNeeds** [A **dataElement**

- that is referenced by a **RoleBasedDataAssignment** (where the attribute **role** is set to **signalBasedDiagnostics**) owned by a **SwcServiceDependency** that aggregates **DiagnosticValueNeeds** that in turn is referenced by a **DiagnosticIoControlNeeds**
- **shall also be referenced** by a **VariableAccess** aggregated in the role **dataSendPoint** by a given **RunnableEntity** that in turn belongs to the enclosing **SwcInternalBehavior**.

- **shall not be referenced** by a `VariableAccess` aggregated in the role `dataWriteAccess` by a given `RunnableEntity` that in turn belongs to the enclosing `SwcInternalBehavior`.

This rule shall be imposed **at the time when the RTE is generated.**]()

In other words, the data handled by the diagnostics transformer shall only be sent explicitly. The usage of **implicit sending of these data is not supported.**

#### 13.8.4.7 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]

#### 13.8.4.8 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]

#### 13.8.4.9 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]

#### 13.8.4.10 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]

#### 13.8.4.11 Dcm Service Use Case: read/write and IOCtrl current values by Client Server Interface

Scenario: an `AtomicSwComponentType` offers a `PPortPrototype` typed by `ClientServerInterface` to read/write and IOCtrl current value via diagnostic services (e.g. measurements, variant coding)

[TPS\_SWCT\_01690] **AtomicSwComponentType** offers a `PPortPrototype` typed by `ClientServerInterface` to read/write and IOCtrl current value via diagnostic services [

**ServiceNeeds kind** [DiagnosticValueNeeds](#), [DiagnosticIoControlNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [DataServices](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

**[TPS\_SWCT\_01691] 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 [SwcServiceDependency](#) that aggregates the [DiagnosticIoControlNeeds](#).]  
([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

### 13.8.4.12 Dcm Service Use Case: A software-component acts as a "file server" to a diagnostic tester

Scenario: an [AtomicSwComponentType](#) acts as a "file server" to a diagnostic tester.

**[TPS\_SWCT\_01791] [AtomicSwComponentType](#) acts as a "file server" to a diagnostic tester** [

**ServiceNeeds kind** [DiagnosticRequestFileTransferNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [RequestFileTransfer](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

<b>Class</b>	<b>DiagnosticRequestFileTransferNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class indicates the existence of a service use case that involves UDS service 0x38, Request File Transfer.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.40: DiagnosticRequestFileTransferNeeds**

### 13.8.5 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.

<b>Class</b>	<b>ObdRatioServiceNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
connectionType	<a href="#">ObdRatioConnectionKindEnum</a>	0..1	attr	Defines how the DEM is connected to the component or module to perform the IUMPR (In use monitor performance ratio) service.
rateBasedMonitoredEvent	<a href="#">DiagnosticEventNeeds</a>	0..1	ref	The rate based monitored Diagnostic Event.
usedFid	<a href="#">FunctionInhibitionNeeds</a>	0..1	ref	This represents the primary Function Inhibition Identifier used for the rate based monitor. This is an optional attribute.

**Table 13.41: ObdRatioServiceNeeds**

<b>Class</b>	<b>ObdControlServiceNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.42: ObdControlServiceNeeds**

<b>Enumeration</b>	<b>ObdRatioConnectionKindEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	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).
<b>Literal</b>	<b>Description</b>
apiUse	The IUMPR service (of the DEM) uses an explicit API to connect to the component or module. <b>Tags:</b> atp.EnumerationLiteralIndex=0
observer	The IUMPR service (of the Dem) uses no API but "observes" the associated diagnostic event. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 13.43: 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.

[constr\_1520] Semantics of **ObdRatioServiceNeeds.rateBasedMonitoredEvent** [In the context of an **SwcServiceDependency**, each **DiagnosticEventNeeds** referenced in the role **rateBasedMonitoredEvent** shall only be referenced by at most a single **ObdRatioServiceNeeds** at the time when the RTE is generated.] ()

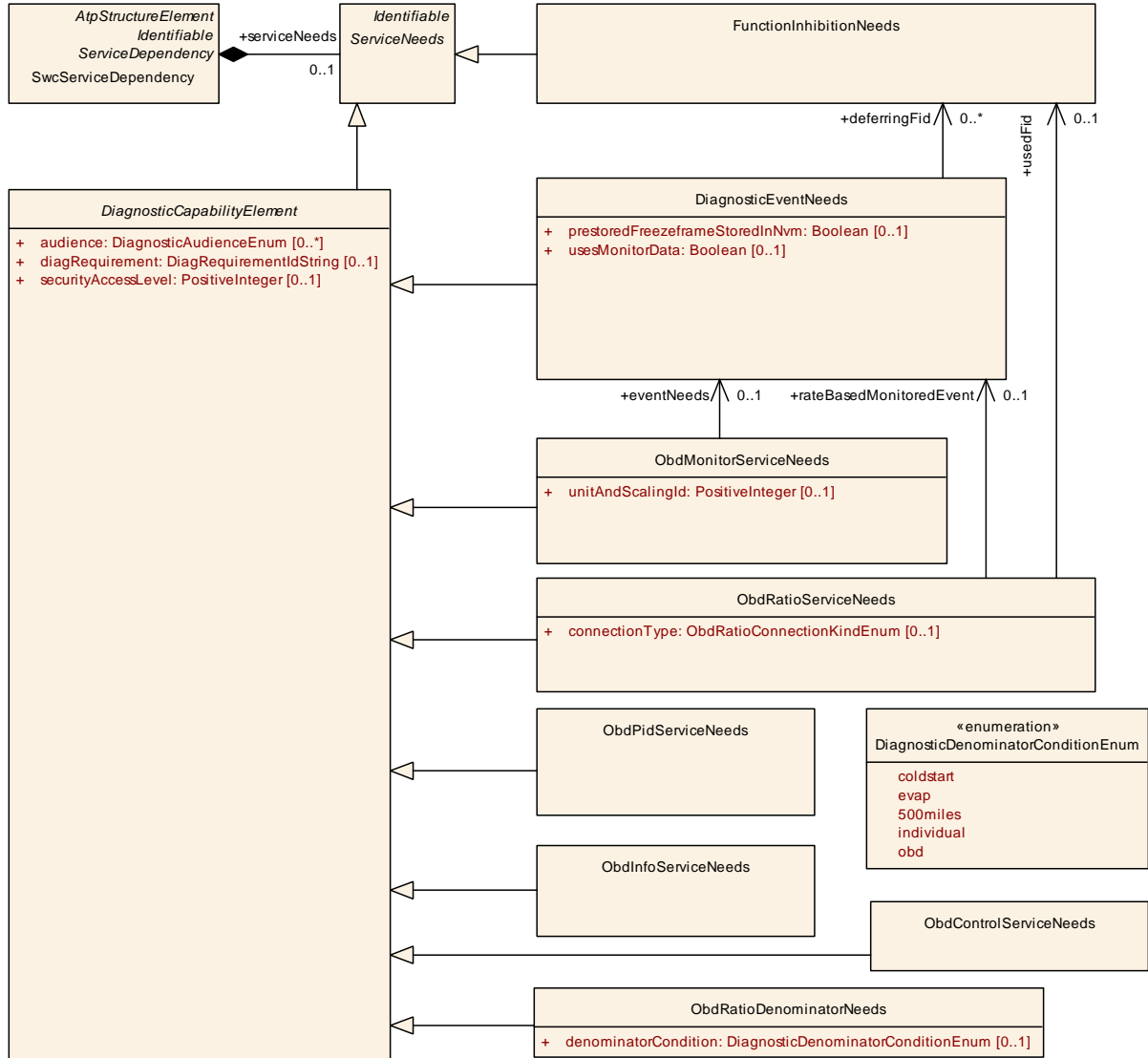


Figure 13.11: **ServiceNeeds**: Diagnostic-related **ServiceNeeds** with emphasis on OBD

<b>Class</b>	<b>ObdPidServiceNeeds</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	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).
<b>Base</b>	ARObject, DiagnosticCapabilityElement, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds





Class	ObdPidServiceNeeds			
Attribute	Type	Mult.	Kind	Note
-	-	-	-	-

**Table 13.44: 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, MultilanguageReferrable, Referrable, ServiceNeeds			
Attribute	Type	Mult.	Kind	Note
-	-	-	-	-

**Table 13.45: 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, MultilanguageReferrable, Referrable, ServiceNeeds			
Attribute	Type	Mult.	Kind	Note
applicationData Type	ApplicationDataType	0..1	ref	reference to an ApplicationDataType that describes the scaling of the data reported by the software-component to the Dem.
eventNeeds	DiagnosticEventNeeds	0..1	ref	This reference identifies the corresponding diagnostic event.
unitAndScaling Id	PositiveInteger	0..1	attr	Unit and scaling ID according to ISO 15031-5.

**Table 13.46: ObdMonitorServiceNeeds**

### 13.8.5.1 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.

This rule shall be imposed **at the time when the RTE is generated.**]()

**13.8.5.2 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].

### 13.8.5.3 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].

Please note that [`constr_1679`] applies for this service use case.

### 13.8.5.4 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].

### 13.8.5.5 Dem Service Use Case: Read DTR data from SW-C for OBD Service \$06

Scenario: an *AtomicSwComponentType* exposes a client port to read DTR value.

[TPS\_SWCT\_02011] **AtomicSwComponentType offers a client port to read DTR value** [

**ServiceNeeds kind** *ObdMonitorServiceNeeds*

**RoleBasedPortAssignment** valid roles:

- DTRCentralReport [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00170, RS\_SWCT\_03190)

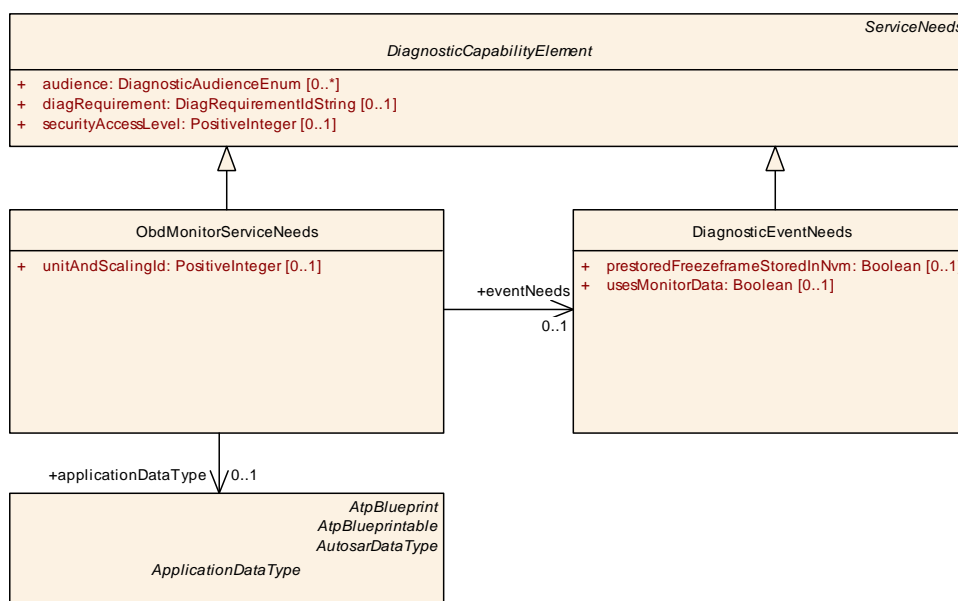


Figure 13.12: Modeling of *ObdMonitorServiceNeeds*

### 13.8.5.6 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].

### 13.8.5.7 Dem Service Use Case: In-Use-Monitoring Performance Ratio Denominator interface

Scenario: an [AtomicSwComponentType](#) implements a denominator (or accesses a ratio for transmission to other control units).

**[TPS\_SWCT\_01765] Dem Service Use Case: In-Use-Monitoring Performance Ratio Denominator interface** [

**ServiceNeeds** kind [ObdRatioDenominatorNeeds](#)

#### RoleBasedPortAssignment

The following roles are applicable:

- [IUMPRDenominatorCondition](#) [1]

#### RoleBasedDataAssignment

N/A

## RepresentedPortGroups

N/A

|(RS\_SWCT\_00170, RS\_SWCT\_03190)

For more information please refer to [SWS\_Dem\_00742].

<b>Class</b>	<b>ObdRatioDenominatorNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class shall be used to indicate that a software-component wants to access the in-use-monitoring performance ration denominator.			
<b>Base</b>	ARObject, DiagnosticCapabilityElement, Identifiable, MultilanguageReferrable, Referrable, Service Needs			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
denominator Condition	DiagnosticDenominatorConditionEnum	0..1	attr	This attribute indicates the applicable denominator condition.

**Table 13.47: ObdRatioDenominatorNeeds**

<b>Enumeration</b>	<b>DiagnosticDenominatorConditionEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	This enumeration contains valid denominator types.
<b>Literal</b>	<b>Description</b>
_500miles	Condition based on definition of 500miles conditions as defined for OBD2. <b>Tags:</b> atp.EnumerationLiteralIndex=2
coldstart	Condition based on definition of "cold start" as defined for EU5+ <b>Tags:</b> atp.EnumerationLiteralIndex=0
evap	Condition based on definition of "EVAP" conditions as defined for OBD2. <b>Tags:</b> atp.EnumerationLiteralIndex=1
individual	condition based on definition of individual requirements. <b>Tags:</b> atp.EnumerationLiteralIndex=3
obd	Condition based on definition of OBD requirements. <b>Tags:</b> atp.EnumerationLiteralIndex=4

**Table 13.48: DiagnosticDenominatorConditionEnum**

### 13.8.6 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 [43].

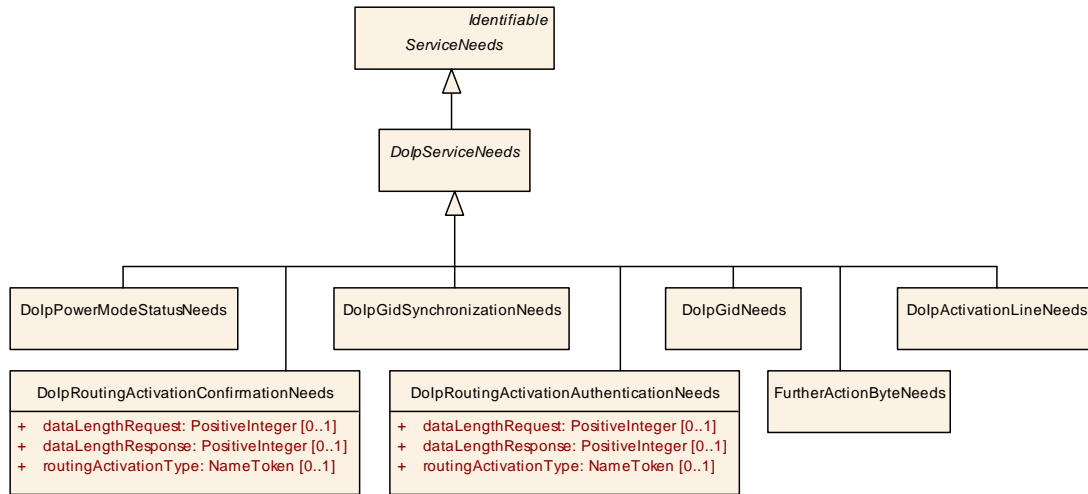


Figure 13.13: Subclasses of **ServiceNeeds** for implementing diagnostics over IP

<b>Class</b>	<b>DolpServiceNeeds</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This represents an abstract base class for ServiceNeeds related to DoIP.			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Subclasses</b>	DolpActivationLineNeeds, DolpGidNeeds, DolpGidSynchronizationNeeds, DolpPowerModeStatusNeeds, DolpRoutingActivationAuthenticationNeeds, DolpRoutingActivationConfirmationNeeds, FurtherActionByteNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

Table 13.49: DolpServiceNeeds

<b>Class</b>	<b>DolpGidNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	ARObject, DolpServiceNeeds, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

Table 13.50: DolpGidNeeds

<b>Class</b>	<b>DolpGidSynchronizationNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	ARObject, DolpServiceNeeds, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–





<b>Class</b>	<b>DolpGidSynchronizationNeeds</b>			
-	-	-	-	-

**Table 13.51: DolpGidSynchronizationNeeds**

<b>Class</b>	<b>DolpPowerModeStatusNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>DolpServiceNeeds</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.52: DolpPowerModeStatusNeeds**

<b>Class</b>	<b>DolpRoutingActivationAuthenticationNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>DolpServiceNeeds</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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 software-component 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 software-component 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.
routing ActivationType	NameToken	0..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 13.53: DolpRoutingActivationAuthenticationNeeds**

<b>Class</b>	<b>DolpRoutingActivationConfirmationNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	DolpRoutingActivationConfirmationNeeds 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.			
<b>Base</b>	<i>ARObject</i> , <i>DolpServiceNeeds</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	DoIpRoutingActivationConfirmationNeeds			
dataLength Request	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 software-component 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 confirmation that is provided by the software entity. If the software entity is a software-component 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.
routing ActivationType	NameToken	0..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 13.54: DoIpRoutingActivationConfirmationNeeds**

Class	DoIpActivationLineNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	A DoIP entity needs to be informed when an external tester is attached or activated. The DoIpActivation ServiceNeeds 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, DoIpServiceNeeds, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
Attribute	Type	Mult.	Kind	Note
-	-	-	-	-

**Table 13.55: DoIpActivationLineNeeds**

### 13.8.6.1 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)

**13.8.6.2 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)

**13.8.6.3 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)

**13.8.6.4 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 `PortInterface` 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)

**13.8.6.5 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`)

#### 13.8.6.6 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 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\_03310, RS\_SWCT\_03190)

<b>Class</b>	<b>WarningIndicatorRequestedBitNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the ability to explicitly request the existence of the WarningIndicatorRequestedBit.			
<b>Base</b>	ARObject, DiagnosticCapabilityElement, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.56: WarningIndicatorRequestedBitNeeds**

### 13.8.6.7 DoIP Service Use Case: Atomic Software-Component provides the further action byte to the DoIP Service Component

Scenario: An AtomicSwComponentType provides the "further action byte" used in vehicle identification/announcement message.

[TPS\_SWCT\_01746] Atomic Software-Component provides the further action byte to the DoIP Service Component [

ServiceNeeds kind FurtherActionByteNeeds

RoleBasedPortAssignment valid roles:

- CallbackGetFurtherActionByte [1]

### RoleBasedDataAssignment

N/A

### RepresentedPortGroups

N/A

|(RS\_SWCT\_03310, RS\_SWCT\_03190)

<b>Class</b>	<b>FurtherActionByteNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	The FurtherActionByteNeeds indicates that the software-component is able to provide the "further action byte" to the DoIP Service Component.			
<b>Base</b>	ARObject, DoIPServiceNeeds, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.57: FurtherActionByteNeeds**

### 13.8.7 Miscellaneous Diagnostic Service Use-Cases

#### 13.8.7.1 Dcm Service Use Case: DiagnosticSessionControl

Scenario: an [AtomicSwComponentType](#) offers an [RPortPrototype](#) typed by a [ModeSwitchInterface](#) to get informed about the diagnostic session.

[TPS\_SWCT\_01706] [AtomicSwComponentType](#) supports *DiagnosticSessionControl* to get informed about the diagnostic session [

**ServiceNeeds** kind [DiagnosticControlNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [Dcm\\_DiagnosticSessionControlModeSwitchInterface](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

The role [Dcm\\_DiagnosticSessionControlModeSwitchInterface](#) is applicable for an [RPortPrototype](#) typed by a [ModeSwitchInterface](#).] ([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

<b>Class</b>	<b>DiagnosticControlNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class indicates a service use-case for reporting the controlled status by diagnostic services.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

Table 13.58: DiagnosticControlNeeds

#### 13.8.7.2 Dcm Service Use Case: EcuReset

Scenario: an [AtomicSwComponentType](#) offers an [RPortPrototype](#) typed by a [ModeSwitchInterface](#) to get informed about the current status of *EcuReset* service.

[TPS\_SWCT\_01707] [AtomicSwComponentType](#) supports *EcuReset* service via diagnostic services [

**ServiceNeeds** kind [DiagnosticControlNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [Dcm\\_EcuResetModeSwitchInterface](#) [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

The role `Dcm_EcuResetModeSwitchInterface` is applicable for an `RPortPrototype` typed by a `ModeSwitchInterface`.] ([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

**13.8.7.3 Dcm Service Use Case: EcuReset ModeRapidPowerShutDown**

Scenario: an `AtomicSwComponentType` offers an `RPortPrototype` typed by a `ModeSwitchInterface` to get informed about the current status of the `EcuReset ModeRapidPowerShutDown` service.

[TPS\_SWCT\_01708] `AtomicSwComponentType` supports `EcuReset ModeRapidPowerShutDown` service via diagnostic services [

`ServiceNeeds` kind `DiagnosticControlNeeds`

**RoleBasedPortAssignment** valid roles:

- `Dcm_ModeRapidPowerShutDownModeSwitchInterface` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

The role `Dcm_ModeRapidPowerShutDownModeSwitchInterface` is applicable for an `RPortPrototype` typed by a `ModeSwitchInterface`.] ([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

**13.8.7.4 Dcm Service Use Case: CommunicationControl**

Scenario: an `AtomicSwComponentType` offers an `RPortPrototype` typed by a `ModeSwitchInterface` to get informed about the current status of the `CommunicationControl` service per ComM Channel.

[TPS\_SWCT\_01709] `AtomicSwComponentType` supports `CommunicationControl` service via diagnostic services [

`ServiceNeeds` kind `DiagnosticControlNeeds`

**RoleBasedPortAssignment** valid roles:

- `Dcm_CommunicationControlModeSwitchInterface` [1]

### RoleBasedDataAssignment

N/A

### RepresentedPortGroups

N/A

The role `Dcm_CommunicationControl` is applicable for an `RPortPrototype` typed by a `ModeSwitchInterface`.] ([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

#### 13.8.7.5 Dcm Service Use Case: ControlDTCSetting

Scenario: an `AtomicSwComponentType` offers an `RPortPrototype` typed by a `ModeSwitchInterface` to get informed about the current status of the `ControlDTC-Setting` service.

**[TPS\_SWCT\_01711] `AtomicSwComponentType` supports `ControlDTCSetting` service via diagnostic services [**

**ServiceNeeds** kind `DiagnosticControlNeeds`

**RoleBasedPortAssignment** valid roles:

- `Dcm_ControlDTCSettingModeSwitchInterface` [1]

### RoleBasedDataAssignment

N/A

### RepresentedPortGroups

N/A

The role `Dcm_ControlDTCSettingModeSwitchInterface` is applicable for an `RPortPrototype` typed by a `ModeSwitchInterface`.] ([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

#### 13.8.7.6 Dcm Service Use Case: Response On Event via diagnostic services

Scenario: an `AtomicSwComponentType` offers a `PPortPrototype` typed by a `ClientServerInterface` as well as an `RPortPrototype` typed by a `ModeSwitchInterface` to support `onChangeOfDataIdentifier Response On Event` (ROE) via diagnostic services.

**[TPS\_SWCT\_02014] `AtomicSwComponentType` supports `Response On Event` (ROE) via diagnostic services [**

**ServiceNeeds** kind `DiagnosticResponseOnEventNeeds`

**RoleBasedPortAssignment** valid roles:

- `Dcm_Roe` [1]

- `Dcm_ResponseOnEventModeSwitchInterface` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

The role `Dcm_ResponseOnEventModeSwitchInterface` is applicable for an `RPortPrototype` typed by a `ModeSwitchInterface` and the role `Dcm_Roe` is applicable for an `PPortPrototype` typed by a `ClientServerInterface`.] ([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

<b>Class</b>	<b>DiagnosticResponseOnEventNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class indicates a service use-case for the diagnostic service ResponseOnEvent.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.59: DiagnosticResponseOnEventNeeds**

**13.8.7.7 Dcm Service Use Case: SecurityAccess**

Scenario: an `AtomicSwComponentType` offers an `RPortPrototype` typed by a `ModeSwitchInterface` to get informed about the current diagnostic security level.

**[TPS\_SWCT\_01712] `AtomicSwComponentType` supports SecurityAccess to get informed about the security level [**

**ServiceNeeds kind** `DiagnosticControlNeeds`

**RoleBasedPortAssignment** valid roles:

- `Dcm_SecurityAccessModeSwitchInterface` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

The role `Dcm_SecurityAccessModeSwitchInterface` is applicable for an `RPortPrototype` typed by a `ModeSwitchInterface`.] ([RS\\_SWCT\\_00170](#), [RS\\_SWCT\\_03190](#))

### 13.8.7.8 Service Use Case: Atomic Software-Component implements a Hardware Shutdown

Scenario: if a hardware component is detected as being defective, the Dem shall inform the `SwComponentPrototype` typed by an `AtomicSwComponentType` which is responsible for executing a hardware-shutdown.

[TPS\_SWCT\_01680] Dem Use Case: Atomic Software-Component implements a Hardware Shutdown [

**ServiceNeeds** kind `DiagnosticComponentNeeds`

**RoleBasedPortAssignment** valid roles:

- `CallbackComponentStatusChanged` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

] (`RS_SWCT_03190`)

<b>Class</b>	<code>DiagnosticComponentNeeds</code>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the ability to specify the service needs for the configuration of component events.			
<b>Base</b>	<code>ARObject</code> , <code>DiagnosticCapabilityElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code> , <code>ServiceNeeds</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

Table 13.60: `DiagnosticComponentNeeds`

### 13.8.7.9 Service Use Case: Upload and download of data

Scenario: a software-component implements the ability to accept data for upload and/or provide data for download. For this purpose the software-component provides a `PPortPrototype` that is supposed to be connected to the Dcm service component.

[TPS\_SWCT\_01769] Dcm Use Case: Upload and download of data [

**ServiceNeeds** kind `DiagnosticUploadDownloadNeeds`

**RoleBasedPortAssignment** valid roles:

- `UploadDownloadServices` [1]

**RoleBasedDataAssignment**

N/A

## RepresentedPortGroups

N/A

|(RS\_SWCT\_03190)

<b>Class</b>	<b>DiagnosticUploadDownloadNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the ability to specify needs regarding upload and download by means of diagnostic services.			
<b>Base</b>	ARObject, <a href="#">DiagnosticCapabilityElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.61: DiagnosticUploadDownloadNeeds**

## 13.9 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 [44]).

<b>Class</b>	<b>DltUserNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	<p>This meta-class specifies the needs on the configuration of the Diagnostic Log and Trace module for one SessionId.</p> <p>This class currently contains no attributes.</p> <p>An instance of this class is used to find out which PortPrototypes of an AtomicSwComponentType belong to this SessionId in order to group the request and response PortPrototypes of the same SessionId.</p> <p>The actual SessionId value is stored in the PortDefinedArgumentValue of the respective PortPrototype specification.</p>			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.62: 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**.



### 13.9.1 Dlt use Case: Application software component transmits debug information

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 Dlt module** [

**ServiceNeeds kind** : [DltUserNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [DltControlService](#) [1]
- [LogTraceSessionControl](#) [1]
- [InjectionCallback](#) [0..1]
- [DltSwcMessageService](#) [0..1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

For more information please refer to [SWS\_Dlt\_00495], [SWS\_Dlt\_00496], and [SWS\_Dlt\_00498].

In this case the software-component has to provide one Client Port (DLTService) in order to register and de-register the context and to send log or trace messages.

## 13.10 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 [45]).

<b>Class</b>	<b>SyncTimeBaseMgrUserNeeds</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
<b>Note</b>	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.
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>





Class	SyncTimeBaseMgrUserNeeds			
Attribute	Type	Mult.	Kind	Note
-	-	-	-	-

**Table 13.63: 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**.

The general idea behind the time synchronization concept is that the role of global time master and global time slave are partly implemented in the application software.

For this purpose, the application software provides `PortPrototypes` typed by the standardized `PortInterfaces` `GlobalTime_Master` and `GlobalTime_Slave`.

In many cases both `PortInterfaces` `GlobalTime_Master` and `GlobalTime_Slave` will be used by the application software of one ECU. This means that the ECU is a global time slave on one domain and a global time master on another domain.

In terms of modeling, a given global time domain is represented by a `SwcServiceDependency`.

If one software-component has to deal with different global time domains (e.g. because it represents a slave in one domain and a master in another) then the corresponding `SwcInternalBehavior` needs to define one `SwcServiceDependency` per global time domain.

### 13.10.1 StbM Use Case: start timer and potentially get notified about its expiration

Scenario: a software-component wants to wind up a timer in the `StbM` with a given expiration time. The software-component may want to receive a notification when the timer expires. In this case the following setup applies:

**[TPS\_SWCT\_01678] StbM use Case: start timer and potentially get notified about its expiration** [

**ServiceNeeds Kind** : `SyncTimeBaseMgrUserNeeds`

**RoleBasedPortAssignment** valid roles:

- StartTimer [1]
- TimeNotification [0..1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

In this case the software-component needs to have an [RPortPrototype](#) typed by the [ClientServerInterface](#) StartTimer and (if applicable) a [PPortPrototype](#) typed by the [ClientServerInterface](#) TimeNotification.

**13.10.2 StbM Use Case: Software-Components wants to get notifications of status changes**

Scenario: a software-component wants to receive events whenever the status of the [StbM](#) changes. For this purpose, the software-component sports a sender/receiver [RPortPrototype](#). In this case the following setup applies:

**[TPS\_SWCT\_01679] StbM use Case: Software-Components wants to get notifications of status changes** [

**ServiceNeeds Kind** : [SyncTimeBaseMgrUserNeeds](#)

**RoleBasedPortAssignment** valid roles:

- StatusNotification [1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

The events received from the [StbM](#) have a fixed structure. For more details, please refer to [SWS\_StbM\_00284].

### 13.10.3 StbM Use Case: Process time snapshot obtained from global time slave for diagnostics purposes

Scenario: a software-component provides a `PPortPrototype` onto which the global time slave pushes snapshots of time synchronization records. This data is typically used for diagnostic purposes.

**[TPS\_SWCT\_01740] StbM use Case: Process time snapshot obtained from global time slave for diagnostics purposes** [

**ServiceNeeds Kind** : `SyncTimeBaseMgrUserNeeds`

**RoleBasedPortAssignment** valid roles:

- `MeasurementNotification` [1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

Note that in this case the software-components acts as a server, the `StbM` implements a client role!

**[TPS\_SWCT\_01741] Suffix used for the resulting name of the `PortInterface` for measurement notification** [The suffix used for the resulting name of the `PortInterface` for the measurement notification `MeasurementNotification_{TB_Name}` shall be taken from the `shortName` of the applicable `SwcServiceDependency`.]([RS\\_SWCT\\_00030](#))

### 13.10.4 StbM Use Case: Software-component represents a global time master

Scenario: a software-component implements the application-software part of the global time master role. For this purpose the software-component exposes an `RPortPrototype` that is supposed to be connected to the `StbM` service component.

**[TPS\_SWCT\_01742] StbM use Case: Software-component represents a global time master** [

**ServiceNeeds Kind** : `SyncTimeBaseMgrUserNeeds`

**RoleBasedPortAssignment** valid roles:

- `GlobalTime_Master` [1]
- `StatusNotification` [0..1]

- MeasurementNotification [0..1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

**[TPS\_SWCT\_01743] Suffix used for the resulting name of the PortInterface for the global time master role** [The suffix used for the resulting name of the PortInterface for the global time master role GlobalTime\_Master\_{Name} shall be taken from the shortName of the applicable SwcServiceDependency.](RS\_SWCT\_00030)

**13.10.5 StbM Use Case: Software-component represents a global time slave**

Scenario: a software-component implements the application-software part of the global time slave role. For this purpose the software-component exposes an RPortPrototype that is supposed to be connected to the StbM service component.

**[TPS\_SWCT\_01744] StbM use Case: Software-component represents a global time slave** [

**ServiceNeeds Kind** : SyncTimeBaseMgrUserNeeds

**RoleBasedPortAssignment** valid roles:

- GlobalTime\_Slave [1]
- StatusNotification [0..1]
- MeasurementNotification [0..1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

]()

[TPS\_SWCT\_01745] Suffix used for the resulting name of the **PortInterface** for the global time slave role [The suffix used for the resulting name of the **PortInterface** for the global time slave role `GlobalTime_Slave_{Name}` shall be taken from the `shortName` of the applicable `SwcServiceDependency`.] (*RS\_SWCT\_00030*)

### 13.10.6 StbM Use Case: Software-component analyzes predictions about the time synchronization process

Scenario: A Time Slave collects information on the time synchronization process, to predict e.g. the sync ingress based on its local instance of the global time and check whether master and slave agree upon the current time. The software-component analyzes the predictions to detect any impairments.

[TPS\_SWCT\_01810] Software-component analyzes predictions about the time synchronization process [

**ServiceNeeds Kind** : `SyncTimeBaseMgrUserNeeds`

**RoleBasedPortAssignment** valid roles:

- `TimeBaseProviderNotification` [1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

] (*RS\_SWCT\_00030*)

For this purpose, the analyzer software-component exposes a `PPortPrototype` typed by the `ClientServerInterface` `TimeBaseProviderNotification_{bus}_{TimeBase}` to the StbM service component.

## 13.11 Secure On-Board Communication

The meta-class `SecureOnBoardCommunicationNeeds` is used together with the `SwcServiceDependency` to define requirements in order to configure the Secure On-Board Communication module (for the terms related to the AUTOSAR Specification of Module `SecOc`, see [46]).

<b>Class</b>	<b>SecureOnBoardCommunicationNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the need for the existence of the SecOc module on the respective ECU. This class currently contains no attributes. An instance of this class is used to find out which ports of a software-component deal with the administration of secure communication in order to group the request and response ports.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
verificationStatusIndicationMode	<a href="#">VerificationStatusIndicationModeEnum</a>	0..1	attr	This attribute provides the ability to control the mode in which the application software is notified about the result of authentication attempts.

**Table 13.64: SecureOnBoardCommunicationNeeds**

<b>Enumeration</b>	<b>VerificationStatusIndicationModeEnum</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This enumeration provides options for setting the mode of a verification status indication.			
<b>Literal</b>	<b>Description</b>			
failureAndSuccess	Verification attempts that came out "false" or "true" shall be forwarded to the application software. <b>Tags:</b> atp.EnumerationLiteralIndex=1			
failureOnly	Only verification attempts that came out "false" shall be forwarded to the application software. <b>Tags:</b> atp.EnumerationLiteralIndex=0			

**Table 13.65: VerificationStatusIndicationModeEnum**

### 13.11.1 SecOc Use Case: obtain the verification status of secure communication

In this scenario, the [ApplicationSwComponentType](#) wants to obtain the status of secure communication.

It is not interested in the details (and would not be able to help anyway even if the details were available).

The [SwcServiceDependency](#) shall aggregate a [SecureOnBoardCommunicationNeeds](#).

**[TPS\_SWCT\_01668] SecOc Use Case: obtain the verification status of secure communication** [

**RoleBasedPortAssignment** valid roles:

n/a

**RoleBasedDataAssignment** valid roles:

- [VerificationStatus](#) [1]

**RepresentedPortGroups**

n/a

]([RS\\_SWCT\\_00030](#))

**[constr\_1681] Existence of attribute `RoleBasedDataAssignment.usedDataElement.localVariable` for `RoleBasedDataAssignment.role = VerificationStatus`** [If the attribute `RoleBasedDataAssignment.role` is set to the value `VerificationStatus` then the reference `RoleBasedDataAssignment.usedDataElement.localVariable` shall not exist **at the time when the RTE is generated.**]()

For explanation of the existence of **[constr\_1681]**, it is not intended to provide access to local variables inside the `SwcInternalBehavior`.

### 13.11.2 SecOc Use Case: software component retires from secure communication for a given period

In this scenario, the `ApplicationSwComponentType` undergoes a reconfiguration period in which it is not able to process any security-related data. During this period, the verification status shall always be set to “failed”.

The `SwcServiceDependency` shall aggregate a `SecureOnBoardCommunicationNeeds`.

**[TPS\_SWCT\_01672] SecOc Use Case: software component retires from secure communication for a given period** [

**RoleBasedPortAssignment** valid roles:

- `VerifyStatusConfiguration` [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

] (*RS\_SWCT\_00030*)

### 13.11.3 SecOc Use Case: deliver freshness to SecOC I

Scenario: a dedicated software-component computes and delivers the freshness to SecOc. The freshness can optionally be truncated by the software-component.

For this purpose, the software-component exposes a `PPortPrototype` to SecOc. This is used for sending a secured message by using the `ClientServerOperation` `GetTxFreshness` or `GetTxFreshnessTruncData`.

**[TPS\_SWCT\_01716] SecOc Use Case: deliver freshness to SecOC I** [

**ServiceNeeds** kind `SecureOnBoardCommunicationNeeds`

**RoleBasedPortAssignment** valid roles:



- FreshnessManagement [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

]([RS\\_SWCT\\_00030](#))

#### 13.11.4 SecOc Use Case: deliver freshness to SecOC II

Scenario: SecOc invokes transmit notification (`SPduTxConfirmation`) at freshness manager. This information can be vital for the computation of the freshness.

[TPS\_SWCT\_01717] [SecOc Use Case: deliver freshness to SecOC II](#) [

**ServiceNeeds** kind [SecureOnBoardCommunicationNeeds](#)

**RoleBasedPortAssignment** valid roles:

- FreshnessManagement [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

]([RS\\_SWCT\\_00030](#))

#### 13.11.5 SecOc Use Case: deliver freshness to SecOC III

Scenario: caused by out-of-sync freshness, SecOC cannot verify a MAC, and contacts the freshness manager (software-component) again to obtain a recalculated freshness value.

Each recalculation of the freshness inside the freshness manager is counted. After a given threshold of retries, SecOC has to drop the received message. For this purpose, the `ClientServerOperation` `GetRxFreshness` or `GetRxFreshnessAuthData` is used.

[TPS\_SWCT\_01718] [SecOc Use Case: deliver freshness to SecOC III](#) [

**ServiceNeeds** kind [SecureOnBoardCommunicationNeeds](#)

**RoleBasedPortAssignment** valid roles:

- FreshnessManagement [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

](RS\_SWCT\_00030)

### 13.11.6 SecOc Use Case: enable the sending of Pdus even if computation of the MAC is not possible

Scenario: there are cases where the ability to send authenticated messages associated with a given freshness id using a default-MAC is required, e.g. if an ECU has been replaced but was not yet provided with cryptographic keys.

Receivers can distinguish this case from the regular authenticated data exchange by looking at the MAC, i.e. a (configurable) default MAC is used in the described case.

[TPS\_SWCT\_01784] **SecOc Use Case: enable the sending of Pdus even if computation of the MAC is not possible** [

**ServiceNeeds** kind [SecureOnBoardCommunicationNeeds](#)

**RoleBasedPortAssignment** valid roles:

- `SendDefaultAuthenticationInformation` [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

](RS\_SWCT\_00030)

### 13.11.7 SecOc Use Case: Receive notification about an authentication attempt

Scenario: a software-component wants to receive a notification about the status of an authentication attempt.

Such a notification can be used to continuously monitor the number of failed verification attempts with the goal to set up a security management system/intrusion detection system that is able to detect an attack that consists of a flood of authentication requests and react with adequate dynamic counter-measures.

For this purpose, the software-component exposes a [PPortPrototype](#) that implements the notification handler. The handler may be interested in only *failed* attempts **or** *failed and passed* attempts.

This aspect can be configured by means of attribute `SecureOnBoardCommunicationNeeds.verificationStatusIndicationMode`.

[TPS\_SWCT\_01832] **SecOc Use Case: Receive notification about an authentication attempt** [

**ServiceNeeds kind** `SecureOnBoardCommunicationNeeds`

**RoleBasedPortAssignment** valid roles:

- `VerificationStatusIndication` [1]

**RoleBasedDataAssignment** valid roles:

n/a

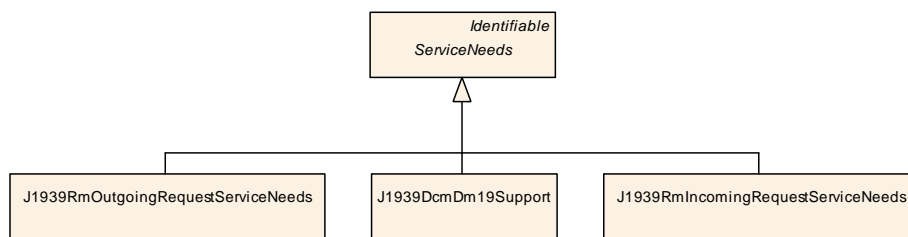
**RepresentedPortGroups**

n/a

] (`RS_SWCT_00030`)

## 13.12 J1939 Communication

The J1939-specific meta-classes `J1939RmOutgoingRequestServiceNeeds` and `J1939RmIncomingRequestServiceNeeds` are used together with the `SwcServiceDependency` to define requirements in order to configure the J1939 request manager (for the terms related to the AUTOSAR Specification of Module **J1939RM**, see [47]).



**Figure 13.14: ServiceNeeds: J1939-related ServiceNeeds**

<b>Class</b>	<b>J1939RmOutgoingRequestServiceNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class shall be used to specify needs with respect to the configuration of the J1939Rm, in particular for the case where an ApplicationSwComponentType needs to send a request to another J1939 node.			
<b>Base</b>	ARObject, <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.66: J1939RmOutgoingRequestServiceNeeds**

<b>Class</b>	<b>J1939RmIncomingRequestServiceNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	"This meta-class shall be used to specify needs with respect to the configuration of the J1939Rm, in particular for the case where an ApplicationSwComponentType needs to accept a request from another J1939 node.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.67: J1939RmIncomingRequestServiceNeeds**

### 13.12.1 J1939RM Use Case: AtomicSwComponentType sends requests to the bus

Scenario: An [AtomicSwComponentType](#) sends requests to the bus. In this case the following setup applies:

To indicate the scenario described in this use case the [SwcServiceDependency](#) shall aggregate a [J1939RmOutgoingRequestServiceNeeds](#).

**[TPS\_SWCT\_01673] Application Software Component sends requests using the J1939Rm** [

**ServiceNeeds kind** [J1939RmOutgoingRequestServiceNeeds](#)

**RoleBasedPortAssignment** valid roles:

- AppSendRequest [1]
- AppAckIndication [1]
- AppRequestTimeoutIndication [0..1]
- AppCancelRequestTimeout [0..1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

]([RS\\_SWCT\\_03180](#))

For more information please refer to the following specification items: [SWS\_-J1939Rm\_00104], [SWS\_J1939Rm\_00106], [SWS\_J1939Rm\_00108], and [SWS\_-J1939Rm\_00105].

### 13.12.2 J1939Rm Use Case: AtomicSwComponentType accepts requests from the bus

Scenario: An [AtomicSwComponentType](#) accepts requests from the bus. In this case the following setup applies:

To indicate the scenario described in this use case the [SwcServiceDependency](#) shall aggregate a [J1939RmIncomingRequestServiceNeeds](#).

**[TPS\_SWCT\_01674] Application Software Component accepts requests using the J1939Rm** [

**ServiceNeeds kind** [J1939RmIncomingRequestServiceNeeds](#)

**RoleBasedPortAssignment** valid roles:

- [AppRequestIndication](#) [1]
- [AppSendAck](#) [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

]([RS\\_SWCT\\_03180](#))

For more information please refer to the following specification items: [SWS\_-J1939Rm\_00103] and [SWS\_J1939Rm\_00107].

### 13.12.3 J1939Dcm wants to retrieve calibration verification numbers from an application software-component

Scenario: J1939Dcm wants to retrieve calibration verification numbers from an application software-component. For this purpose, the software-component exposes a [PPortPrototype](#) to the J1939Dcm

**[TPS\_SWCT\_01809] J1939Dcm wants to retrieve calibration verification numbers from an application software-component** [

**ServiceNeeds kind** [J1939DcmDm19Support](#)

**RoleBasedPortAssignment** valid roles:

- [J1939Dcm\\_CalibrationInformation](#) [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

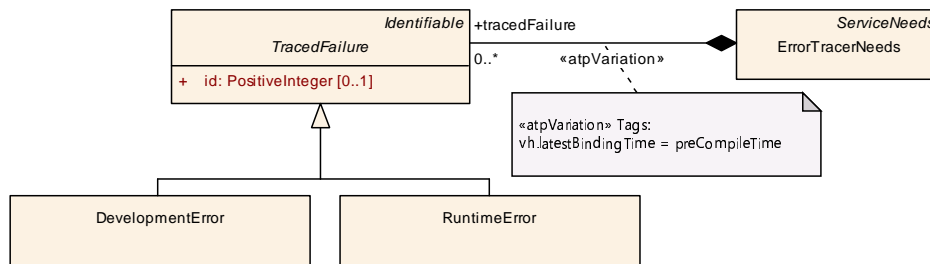
|(RS\_SWCT\_03180)

<b>Class</b>	<b>J1939DcmDm19Support</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	The software-component provides information about calibration verification numbers for inclusion in DM19			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.68: J1939DcmDm19Support**

### 13.13 Error Tracer

The meta-class [ErrorTracerNeeds](#) is used to define requirements in order to configure the Default Error Tracer.



**Figure 13.15: Modeling of [ErrorTracerNeeds](#)**

In particular, [ErrorTracerNeeds](#) provides the exhaustive list of all [tracedFailure](#) implemented in the enclosing software-component and reported via the [PortPrototype](#) referenced via [RoleBasedPortAssignment](#).

Each [tracedFailure](#) relates to one ID, represented by attribute [TracedFailure.id](#).

For more explanation, please consult with the specification of Default Error Tracer [48].

<b>Class</b>	<b>ErrorTracerNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the need to report failures to the error tracer.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
tracedFailure	<a href="#">TracedFailure</a>	*	aggr	list of traced failures <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table 13.69: ErrorTracerNeeds**

<b>Class</b>	<b>TracedFailure</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the ability to report a specific failure to the error tracer. The short name specifies the literal applicable for the Default Error Tracer.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">DevelopmentError</a> , <a href="#">RuntimeError</a> , <a href="#">TransientFault</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
id	PositiveInteger	0..1	attr	ID of detected failure used in reporting API as error or fault id.

**Table 13.70: TracedFailure**

<b>Class</b>	<b>DevelopmentError</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	The reported failure is classified as development error.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">TracedFailure</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.71: DevelopmentError**

<b>Class</b>	<b>RuntimeError</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	The reported failure is classified as runtime error.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">TracedFailure</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.72: RuntimeError**

### 13.13.1 Error Tracer Use Case: Default Error Tracer Service use Case: report failure

#### [TPS\_SWCT\_01694] Setup for Default Error Tracer Service use Case: development errors or runtime error [

Scenario: a software-component reports development errors or runtime error to the Default Error Tracer. In this case the following setup applies

**ServiceNeeds** kind [ErrorTracerNeeds](#)

**RoleBasedPortAssignment** valid roles:

- DETService[1]

**RoleBasedDataAssignment** valid roles:

n/a

**RoleBasedDataTypeAssignment** valid roles:

n/a

**RepresentedPortGroups**

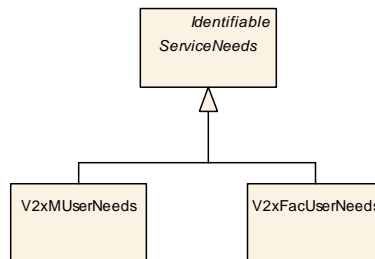
n/a

](RS\_SWCT\_00030)

[constr\_1433] **Transient faults are not applicable to software-components** [An `ErrorTracerNeeds` aggregated in the context of a `SwcInternalBehavior` is - **at the time when the RTE is generated** - not allowed to own a `Transient-Fault` in the role `ErrorTracerNeeds.tracedFailure`.]()

### 13.14 Vehicle-2-X Facilities

The meta-class `V2xFacUserNeeds` is used together with the `SwcServiceDependency` to define requirements in order to configure the V2xFac module (for the terms related to the AUTOSAR Specification of Module V2xFac see [49]).



**Figure 13.16: Modeling of V2xFacUserNeeds**

<b>Class</b>	<b>V2xFacUserNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the ability to define service needs for V2x facilities.			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.73: V2xFacUserNeeds**

Please note that for the described use cases of the V2xFac Service following rule applies:

For every used `SenderReceiverInterface` it is necessary to create a `RoleBasedDataAssignment`.



### 13.14.1 V2xFac Use Case: Application software component provides Vehicle specific data to the V2X-Stack for CAM transmission

Scenario: an [AtomicSwComponentType](#) autonomously calls the V2xFac, providing vehicle data collected via the in-vehicle networks in the module. In this case the following setup applies:

**[TPS\_SWCT\_01728] V2xFac Use Case: Application software component provides Vehicle specific data to the V2X-Stack for CAM transmission** [

**RoleBasedPortAssignment**

N/A

**RoleBasedDataAssignment** valid roles:

- V2xFacVdp [1]

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

In this case the software component has to provide one Sender Port (V2xFacVdp) to

- provide the Vehicle Data

**[constr\_1682] Existence of attribute [RoleBasedDataAssignment.usedDataElement.localVariable](#) for [RoleBasedDataAssignment.role = V2xFacVdp](#)** [If the attribute [RoleBasedDataAssignment.role](#) is set to the value V2xFacVdp then the reference [RoleBasedDataAssignment.usedDataElement.localVariable](#) shall not exist **at the time when the RTE is generated.** ]()

For explanation of the existence of [\[constr\\_1682\]](#), it is not intended to provide access to local variables inside the [SwcInternalBehavior](#).

### 13.14.2 V2xFac Use Case: V2xFac notifies application software component about received messages

Scenario: an [AtomicSwComponentType](#) shall be informed by the V2xFac about received CAM or DENM messages. In this case the following setup applies:

**[TPS\_SWCT\_01729] V2xFac Use Case: V2xFac notifies application software component about received messages** [

**RoleBasedPortAssignment**

N/A

**RoleBasedDataAssignment** valid roles:

- V2xApplRxIndicationCam [0..1]
- V2xApplRxIndicationDenm [0..1]

### RoleBasedDataTypeAssignment

N/A

### RepresentedPortGroups

N/A

|(RS\_SWCT\_00030)

In this case the software-component has to provide one Receiver Port (V2xApplRxIndicationCam) to receive the current value of the CAM message and / or one Receiver Port (V2xApplRxIndicationDenm) to receive the current DENM message.

Please note that at least one of the two possible [RoleBasedDataAssignments](#) shall exist for this use case.

**[constr\_1683] Existence of attribute [RoleBasedDataAssignment.usedDataElement.localVariable](#) for [RoleBasedDataAssignment.role = V2xApplRxIndicationCam](#)** [If the attribute [RoleBasedDataAssignment.role](#) is set to the value V2xApplRxIndicationCam then the reference [RoleBasedDataAssignment.usedDataElement.localVariable](#) shall not exist **at the time when the RTE is generated.**]()

For explanation of the existence of [\[constr\\_1683\]](#), it is not intended to provide access to local variables inside the [SwcInternalBehavior](#).

### 13.14.3 V2xFac Use Case: Application software component triggers transmission of DENM message

Scenario: an [AtomicSwComponentType](#) shall be able to trigger the transmission of different DENM types. In this case the following setup applies:

**[TPS\_SWCT\_01730] V2xFac Use Case: Application software component triggers transmission of DENM message** [

**RoleBasedPortAssignment** valid roles:

- V2xFacDenBs [1]

### RoleBasedDataAssignment

N/A

### RoleBasedDataTypeAssignment

N/A

### RepresentedPortGroups

N/A

](RS\_SWCT\_00030)

In this case the software component has to provide one Client `PortPrototype` (typed by the standardized `ClientServerInterface V2xFacDenBs`) to

- trigger new DENM message
- trigger updated DENM message
- trigger a cancellation of a DENM message

#### 13.14.4 V2xFac Use Case: Application software component processes the *MAP (topology) Extended Message*

Scenario: an `AtomicSwComponentType` shall be able to process the *MAP (topology) Extended Message*. In this case the following setup applies:

**[TPS\_SWCT\_01764] V2xFac Use Case: Application software component shall be able to process the *MAP (topology) Extended Message* [**

**RoleBasedPortAssignment**

N/A

**RoleBasedDataAssignment** valid roles:

- `V2xApplRxIndicationMapem [1]`

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

In this case the software component has to provide one `RPortPrototype` (typed by the standardized `SenderReceiverInterface V2xApplRxIndicationMapem`) to process the *MAP (topology) Extended Message (MAPEM)*.

**[constr\_1684] Existence of attribute `RoleBasedDataAssignment.usedDataElement.localVariable` for `RoleBasedDataAssignment.role = V2xApplRxIndicationMapem` [If the attribute `RoleBasedDataAssignment.role` is set to the value `V2xApplRxIndicationMapem` then the reference `RoleBasedDataAssignment.usedDataElement.localVariable` shall not exist at the time when the RTE is generated.]()**

For explanation of the existence of `[constr_1684]`, it is not intended to provide access to local variables inside the `SwcInternalBehavior`.

### 13.14.5 V2xFac Use Case: Application software component processes Infrastructure to Vehicle Information Message

Scenario: an `AtomicSwComponentType` shall be able to process the *Infrastructure to Vehicle Information Message* (IVIM). In this case the following setup applies:

**[TPS\_SWCT\_01770] V2xFac Use Case: Application software component processes Infrastructure to Vehicle Information Message** [

**RoleBasedPortAssignment**

N/A

**RoleBasedDataAssignment** valid roles:

- `V2xApplRxIndicationIvim` [1]

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

](*RS\_SWCT\_00030*)

In this case the software component has to provide one `RPortPrototype` (typed by the standardized `SenderReceiverInterface` `V2xApplRxIndicationIvim`) to process the *Infrastructure to Vehicle Information Message* (IVIM).

**[constr\_1685] Existence of attribute `RoleBasedDataAssignment.usedDataElement.localVariable` for `RoleBasedDataAssignment.role = V2xApplRxIndicationIvim`** [If the attribute `RoleBasedDataAssignment.role` is set to the value `V2xApplRxIndicationIvim` then the reference `RoleBasedDataAssignment.usedDataElement.localVariable` shall not exist **at the time when the RTE is generated.** ]()

For explanation of the existence of [constr\_1685], it is not intended to provide access to local variables inside the `SwcInternalBehavior`.

### 13.14.6 V2xFac Use Case: Application software component processes Signal Phase And Timing Extended Message

Scenario: an `AtomicSwComponentType` shall be able to process the *Signal Phase And Timing Extended Message* (SPATEM). In this case the following setup applies:

**[TPS\_SWCT\_01788] V2xFac Use Case: Application software component processes Signal Phase And Timing Extended Message** [

**RoleBasedPortAssignment**

N/A

**RoleBasedDataAssignment** valid roles:

- V2xApplRxIndicationSpatem [1]

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

|(RS\_SWCT\_00030)

In this case the software component has to provide one RPortPrototype (typed by the standardized SenderReceiverInterface V2xApplRxIndicationSpatem) to process the *Signal Phase And Timing Extended Message* (SPATEM).

**[constr\_1686] Existence of attribute RoleBasedDataAssignment.usedDataElement.localVariable for RoleBasedDataAssignment.role = V2xApplRxIndicationSpatem** [If the attribute RoleBasedDataAssignment.role is set to the value V2xApplRxIndicationSpatem then the reference RoleBasedDataAssignment.usedDataElement.localVariable shall not exist at the time when the RTE is generated.]()

For explanation of the existence of [constr\_1686], it is not intended to provide access to local variables inside the SwcInternalBehavior.

### 13.15 Vehicle-2-X Management

The meta-class V2xMUserNeeds is used together with the SwcServiceDependency to define requirements in order to configure the V2X Manager module (for the terms related to the AUTOSAR Specification of Module V2xM see [50]).

<b>Class</b>	<b>V2xMUserNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the ability to express service needs for the V2x management.			
<b>Base</b>	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
-	-	-	-	-

**Table 13.74: V2xMUserNeeds**

Please note that for the described use cases of the V2xFac 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**.

### 13.15.1 V2xM Use Case: Application software component provides Vehicle specific data to the V2X-Stack for Position and Time information

Scenario: an [AtomicSwComponentType](#) autonomously calls the V2X Manager, providing vehicle data collected via the in-vehicle networks in the module. In this case the following setup applies:

#### [TPS\_SWCT\_01731] V2xM Use Case: Application software component provides Vehicle specific data to the V2X-Stack for Position and Time information [

**RoleBasedPortAssignment** valid roles:

- V2xM\_Vdp [1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

In this case the software component has to provide one Client [PortPrototype](#) (typed by the standardized [ClientServerInterface](#) V2xM\_Vdp) to

- set the current time and position

### 13.15.2 V2xM Use Case: Application software component needs V2X specific data from the V2X Manager

Scenario: an [AtomicSwComponentType](#) autonomously calls the V2X Manager, getting information of V2X specific data in the module. In this case the following setup applies:

#### [TPS\_SWCT\_01732] V2xM Use Case: Application software component needs V2X specific data from the V2X Manager [

**RoleBasedPortAssignment** valid roles:

- V2xM\_Vdp [1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

In this case the software component has to provide one Client `PortPrototype` (typed by the standardized `ClientServerInterface V2xM_Vdp`) to

- access the current time of the V2X-Stack, based on the system clock
- access the earliest date of expiration of a Long Term Certificate
- access the earliest date of expiration of a Pseudonym Certificate

### 13.15.3 V2xM Use Case: Application software component has soft-control over Pseudonym-Change within V2X Manager

Scenario: an `AtomicSwComponentType` autonomously calls the V2X Manager, setting the locked or unlocked state for pseudonym change. In this case the following setup applies:

**[TPS\_SWCT\_01733] V2xM Use Case: Application software component has soft-control over Pseudonym-Change within V2X Manager** [

**RoleBasedPortAssignment** valid roles:

- `V2xM_PseudonymChange` [1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

](RS\_SWCT\_00030)

In this case the software component has to provide one Client `PortPrototype` (typed by the standardized `ClientServerInterface V2xM_PseudonymChange`) to

- set the state of the pseudonym change to locked
- set the state of the pseudonym change to unlocked

### 13.15.4 V2xM Use Case: Application software component has the ability to do Verification-on-Demand

Scenario: an `AtomicSwComponentType` autonomously calls the V2X Manager, authenticating a previously received message. In this case the following setup applies:

**[TPS\_SWCT\_01734] V2xM Use Case: Application software component has the ability to do Verification-on-Demand** [

**RoleBasedPortAssignment** valid roles:

- V2xM\_Sec [1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

In this case the software component has to provide one Client [PortPrototype](#) (typed by the standardized [ClientServerInterface](#) V2xM\_Sec) to

- verify a previously received message

### 13.15.5 V2xM Use Case: Application software component do location based calculations

Scenario: an [AtomicSwComponentType](#) autonomously calls the V2X Manager, getting results for geographical calculations. In this case the following setup applies:

**[TPS\_SWCT\_01735] V2xM Use Case: Application software component do location based calculations** [

**RoleBasedPortAssignment** valid roles:

- V2xM\_GeoMath [1]

**RoleBasedDataAssignment**

N/A

**RoleBasedDataTypeAssignment**

N/A

**RepresentedPortGroups**

N/A

]([RS\\_SWCT\\_00030](#))

In this case the software component has to provide one Client [PortPrototype](#) (typed by the standardized [ClientServerInterface](#) V2xM\_GeoMath) to

- calculate the distance between two location tuples (latitude, longitude)
- calculate an allowed tolerance value between two heading values



## 13.16 Hardware Test Manager

The service use cases for the *Hardware Test Manager* are indicated by the usage of meta-class `HardwareTestNeeds` in the role `SwcServiceDependency.serviceNeeds`.

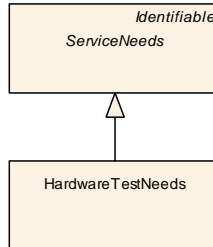


Figure 13.17: Modeling of `HardwareTestNeeds`

<b>Class</b>	<code>HardwareTestNeeds</code>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class represents the ability to indicate that a software-component is interested in the results of the hardware test and will establish a <code>PortPrototype</code> to query the hardware test manager.			
<b>Base</b>	<code>ARObject</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code> , <code>ServiceNeeds</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

Table 13.75: `HardwareTestNeeds`

### 13.16.1 HtssM Service Use Case: Query results of hardware tests

Scenario: A software-component wants to query the results of hardware tests conducted by the `HtssM`. For this purpose, the software-component exposes an `RPortPrototype` that shall be connected to the `HtssM`.

[TPS\_SWCT\_01763] **HtssM Service Use Case: Query results of hardware tests** [

**ServiceNeeds kind** : `HardwareTestNeeds`

**RoleBasedPortAssignment** valid roles:

- `GetTestStatus` [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups**

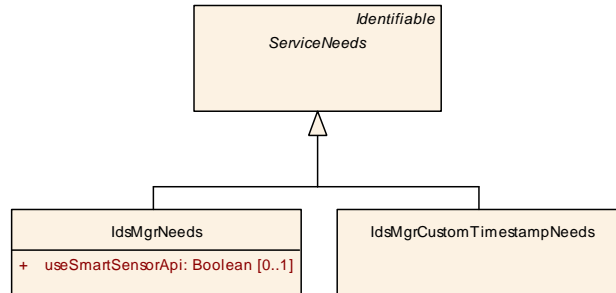
N/A

]([RS\\_SWCT\\_00030](#))

### 13.17 Intrusion Detection System Manager

The interaction with the Intrusion Detection System Manager consists of the ability to report security events.

For this purpose, two different kinds of *ServiceNeeds* are provided: *IdsMgrNeeds* for the actual reporting of security events, and *IdsMgrCustomTimestampNeeds* for the retrieval of a timestamp by the IdsM.



**Figure 13.18: Modeling of *ServiceNeeds* for intrusion detection system management**

<b>Class</b>	<b>IdsMgrNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class is used to indicate that the enclosing SwcServiceDependency represents a service use case for the Intrusion Detection System Manager. <b>Tags:</b> atp.Status=draft			
<b>Base</b>	ARObject, <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
useSmartSensorApi	Boolean	0..1	attr	This attribute controls whether the reporting of the security event shall be done by means of the smart sensor API.

**Table 13.76: IdsMgrNeeds**

<b>Class</b>	<b>IdsMgrCustomTimestampNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	This meta-class is used to indicate that the enclosing SwcServiceDependency represents a service use case for the retrieval of a custom timestamp by the Intrusion Detection System Manager. <b>Tags:</b> atp.Status=draft			
<b>Base</b>	ARObject, <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 13.77: IdsMgrCustomTimestampNeeds**

#### 13.17.1 IdsM Service Use Case: AtomicSwComponentType reports security event

Scenario: An *AtomicSwComponentType* reports a security event. It does not use the so-called Smart Sensor API.

To indicate the scenario described in this use case the `SwcServiceDependency` shall aggregate a `IdsMgrNeeds`.

**[TPS\_SWCT\_01826]{DRAFT} Application Software Component reports security event** [

**ServiceNeeds** kind `IdsMgrNeeds`

**RoleBasedPortAssignment** valid roles:

- `IdsMService` [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

](*RS\_SWCT\_00030*)

**[TPS\_SWCT\_01827]{DRAFT} Suffix used for the resulting name of the `PortInterface` for the `IdsM` Services** [The *suffix* used for the resulting name of the `PortInterface` for the Data Services (`IdsMService_{EventName}`) shall be taken from the `shortName` of the applicable `SwcServiceDependency`.] (*RS\_SWCT\_00030*)

### 13.17.2 `IdsM` Service Use Case: `AtomicSwComponentType` reports security event using Smart Sensor API

Scenario: An `AtomicSwComponentType` reports a security event. For this purpose, the so-called `Smart Sensor API` is used.

To indicate the scenario described in this use case the `SwcServiceDependency` shall aggregate a `IdsMgrNeeds`.

**[TPS\_SWCT\_01828]{DRAFT} Application Software Component reports security event using Smart Sensor API** [

**ServiceNeeds** kind `IdsMgrNeeds`

**RoleBasedPortAssignment** valid roles:

- `IdsMSmartSensorService` [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

n/a

]()

**[TPS\_SWCT\_01829]{DRAFT} Suffix used for the resulting name of the `PortInterface` for the `IdsM` Services** [The *suffix* used for the resulting name of

the `PortInterface` for the Data Services (`IdsMSmartSensorService_{EventName}`) shall be taken from the `shortName` of the applicable `SwcServiceDependency`.] (*RS\_SWCT\_00030*)

### 13.17.3 IdsM Service Use Case: `AtomicSwComponentType` provides time stamp to IdsM

Scenario: An `AtomicSwComponentType` exposes a `PPortPrototype` for providing a time stamp to the IdsM.

To indicate the scenario described in this use case the `SwcServiceDependency` shall aggregate a `IdsMgrCustomTimestampNeeds`.

[TPS\_SWCT\_01830]{DRAFT} **Application Software Component provides time stamp to IdsM** [

**ServiceNeeds** kind `IdsMgrCustomTimestampNeeds`

**RoleBasedPortAssignment** valid roles:

- `IdsMCustomTimestamp` [1]

**RoleBasedDataAssignment** valid roles:

n/a

**RepresentedPortGroups**

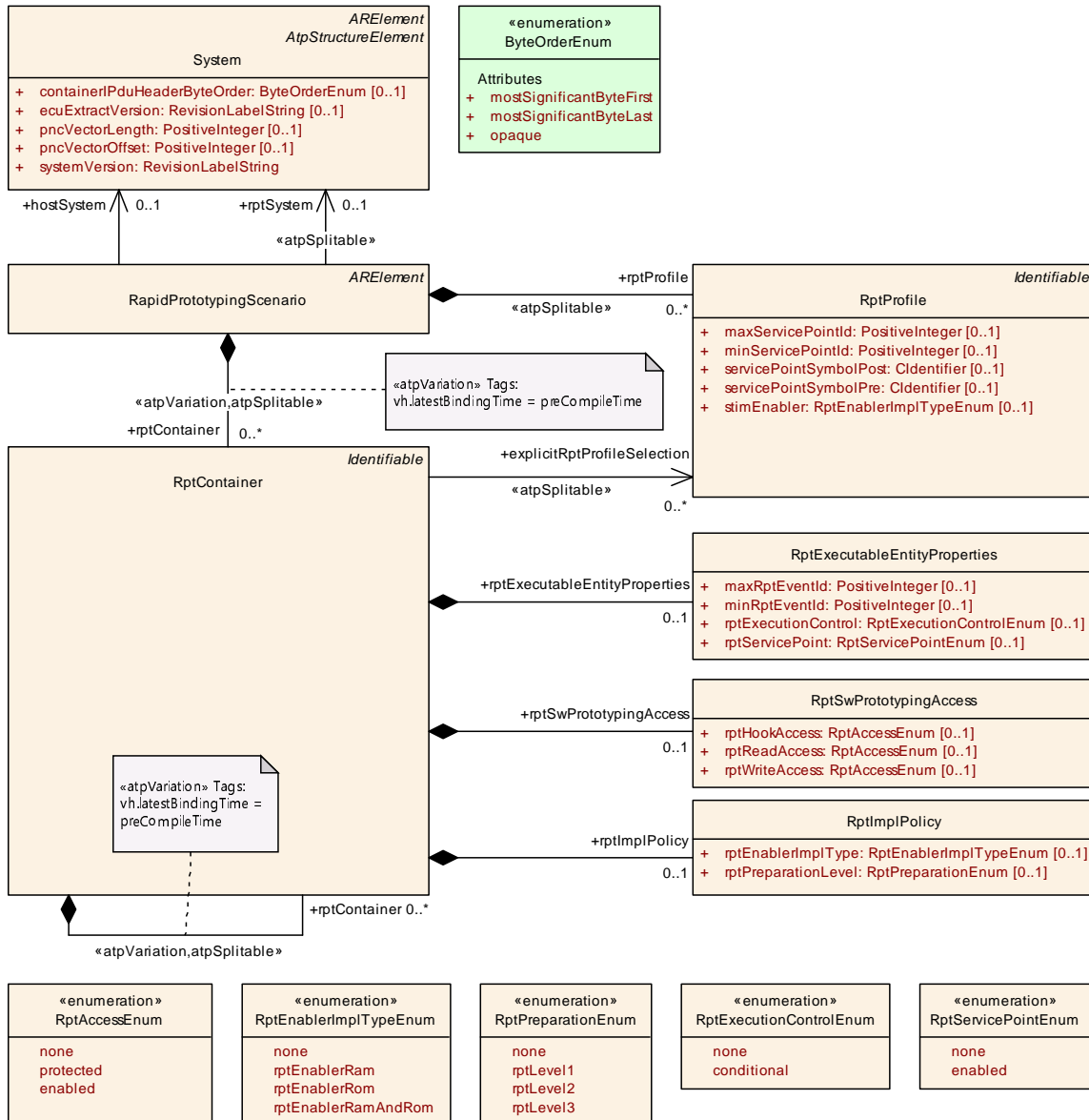
n/a

] (*RS\_SWCT\_00030*)

# 14 Rapid Prototyping Scenarios

## 14.1 Definition of Rapid Prototyping Scenario

A Rapid Prototyping Scenario consist out of two main aspects: The description of the `byPassPoints` (see Figure 14.1) and the relation to a `rptHook` (see Figure 14.2).



**Figure 14.1: Rapid Prototyping Scenario**

A Rapid Prototyping Scenario is structured by means of `RptContainers`. The correct usage of `RptContainer` structure is described in 14.2.

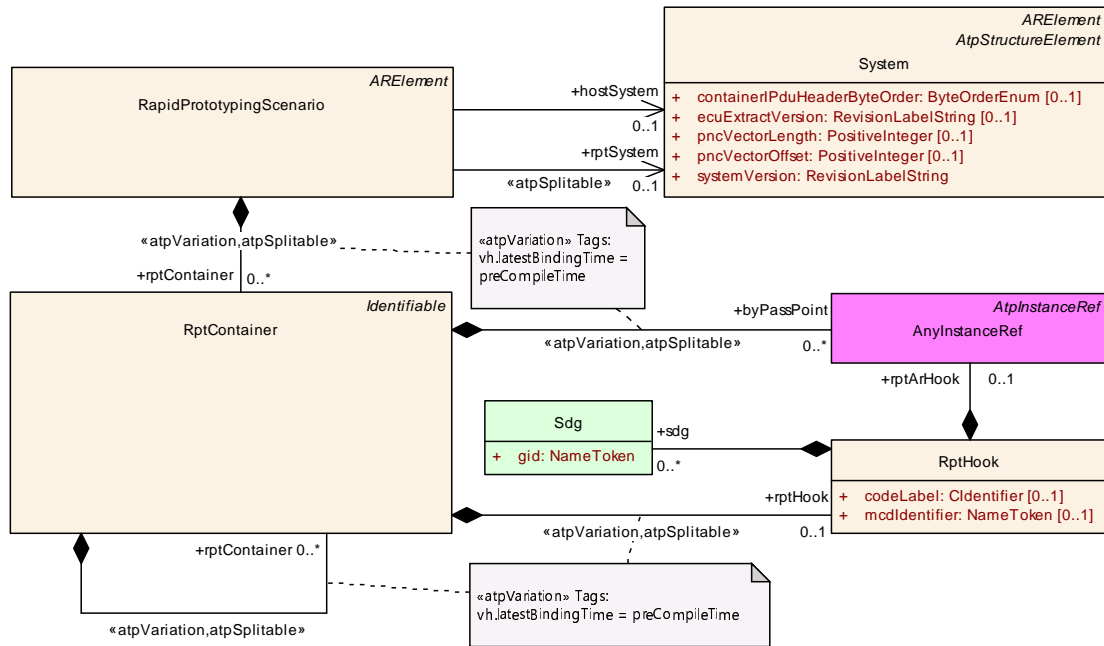


Figure 14.2: Rapid Prototyping Hooks

<b>Class</b>	<b>RapidPrototypingScenario</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
<b>Note</b>	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 rpt Hook. <b>Tags:</b> atp.recommendedPackage=RapidPrototypingScenarios			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
hostSystem	<a href="#">System</a>	0..1	ref	System which describes the software components of the host ECU.
rptContainer	<a href="#">RptContainer</a>	*	aggr	Top-level rptContainer definitions of this specific rapid prototyping scenario. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=rptContainer.shortName, rpt Container.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
rptProfile	<a href="#">RptProfile</a>	*	aggr	Defiens the applicable Rapid Prototyping profils which are especially defining the smbol of the service functions and the valid id range. The order of the RptProfiles determines the order of the service function invocation by RTE. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=rptProfile.shortName
rptSystem	<a href="#">System</a>	0..1	ref	System which describes the rapid prototyping algorithm in the format of AUTOSAR Software Components. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=rptSystem

Table 14.1: RapidPrototypingScenario

[constr\_1987] Existence of instance reference [RapidPrototypingScenario](#). [hostSystem](#) [For each [RapidPrototypingScenario](#), the instance reference to

ModeDeclaration in the role hostSystem shall exist at the time when the RTE is generated.>()

Class	RptContainer			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	<p>This meta-class defines a byPassPoint and the relation to a rptHook.</p> <p>Additionally it may contain further rptContainers if the byPassPoint is not atomic. For example a byPassPoint referencing to a RunnableEntity may contain rptContainers referring to the data access points of the RunnableEntity.</p> <p>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.</p>			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Type	Mult.	Kind	Note
byPassPoint	AtpFeature	*	iref	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=byPassPoint.contextElement, byPassPoint.target, byPassPoint.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime  <b>InstanceRef implemented by:</b>AnyInstanceRef</p>
explicitRptProfileSelection	RptProfile	*	ref	<p>This attribute defines the applicable RptProfiles for the specific RptContainer. If not any references to a specific RptProfile is defined, all RptProfiles defined in the Rapid PrototypingScenario are applicable.</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>atp.Splitkey=explicitRptProfileSelection</p>
rptContainer	RptContainer	*	aggr	<p>Sub-level rptContainer definitions of this specific rapid prototyping scenario.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=rptContainer.shortName, rptContainer.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime</p>
rptExecutableEntityProperties	RptExecutableEntityProperties	0..1	aggr	<p>Describes the required code preparation for rapid prototyping at ExecutableEntity invocation.</p>
rptHook	RptHook	0..1	aggr	<p>The rptHook describes the link between a byPassPoint and the rapid prototyping algorithm.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=rptHook, rptHook.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime</p>
rptImplPolicy	RptImplPolicy	0..1	aggr	<p>Describes the required code preparation for rapid prototyping at data accesses.</p>
rptSwPrototypingAccess	RptSwPrototypingAccess	0..1	aggr	<p>Describes the required accessibility of data and modes by the rapid prototyping tooling.</p>

Table 14.2: RptContainer

<b>Class</b>	<b>RptHook</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
codeLabel	CIdentifier	0..1	attr	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. <b>InstanceRef implemented by:</b> <a href="#">AnyInstanceRef</a>
sdg	<a href="#">Sdg</a>	*	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 14.3: 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 an 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](#), [RS\\_SWCT\\_03282](#))

**[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 Software Component the [rptArHook](#) reference is applicable. Otherwise, the definition of a [codeLabel](#) and optionally [mcdIdentifier](#) shall be used.] ([RS\\_SWCT\\_03280](#), [RS\\_SWCT\\_03281](#), [RS\\_SWCT\\_03282](#))

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 [rpt-System](#) shall be of [category](#) RPT\_SYSTEM.

This rule shall be imposed **at the time when the RTE is generated.**]()

## 14.2 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 a 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 14.4: 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 14.4.

This rule shall be imposed **at the time when the RTE is generated.**]()

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 selection of access points 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`.

This rule shall be imposed **at the time when the RTE is generated.**]()

### 14.3 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** [The aggregations in the roles `rptContainer`, `byPassPoint`, `explicitRptProfileSelection`, and `rptHook` at `RptContainer` are decorated by stereotype `<<atpSplitable>>`. By this means, it is possible to specify aspects of `RptContainers` 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 relationship to lower level elements in a component description, such as `VariableAccess`, where in contrast the `byPassPoints` may only be specified on higher level elements such as `SwComponentPrototypes` in a first step.

### 14.4 Modifications of the Meta-Model for supporting the RPT scenario

The implementation of the rapid-prototyping scenario implies the definition of *access points* (see table 14.4). 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 14.3). Two meta-classes in this collection, however, are not derived from `Referrable`:

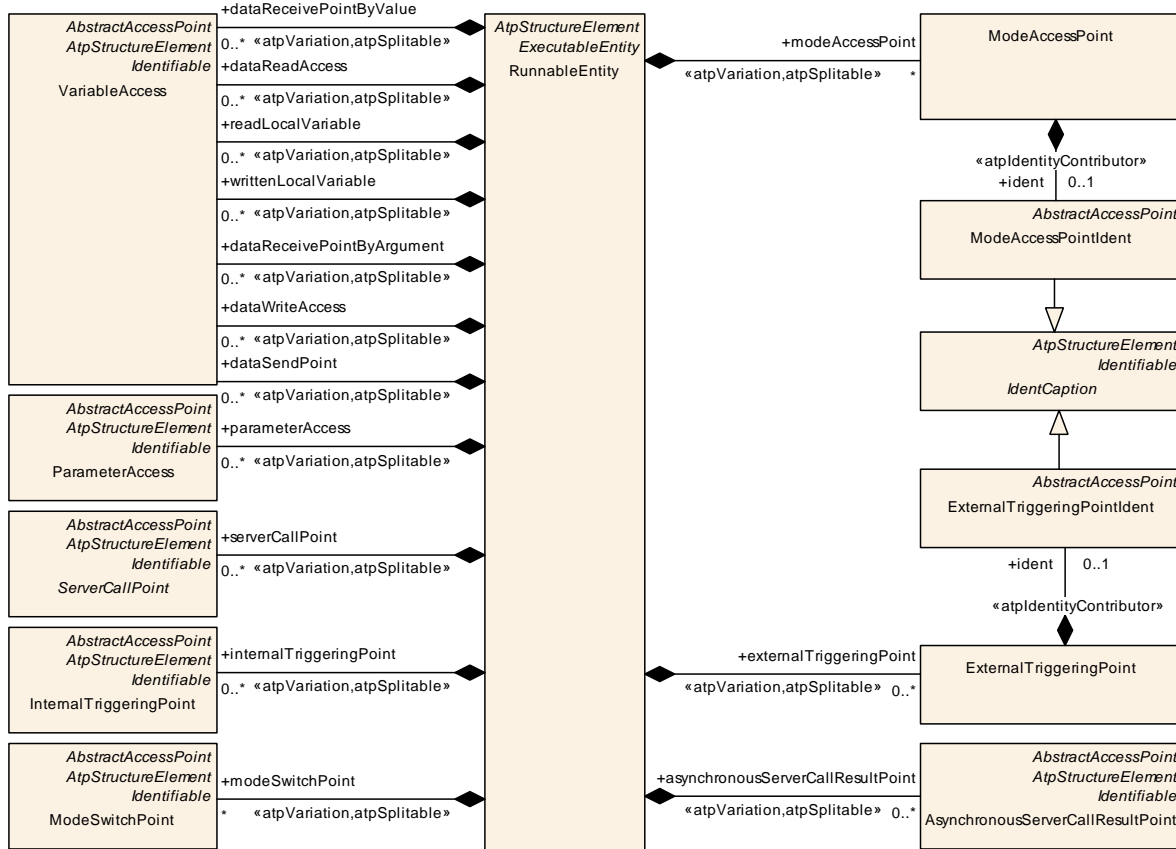
- `ExternalTriggeringPoint`
- `ModeAccessPoint`

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 Schema<sup>1</sup>.

---

<sup>1</sup>Because in this case the `shortName` becomes mandatory.

Therefore, a different approach (as sketched in Figure 14.3) has been implemented.



**Figure 14.3: 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**<sup>2</sup> aggregated in the role `ident` by `ModeAccessPoint`, or in the role `ident` by meta-class `ExternalTriggeringPoint`.

<b>Class</b>	<i>IdentCaption</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
<b>Note</b>	This meta-class represents the caption. This allows having some meta-classes optionally identifiable.			
<b>Base</b>	<i>ARObject</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Subclasses</b>	<i>BswServiceDependencyIdent</i> , <i>ExternalTriggeringPointIdent</i> , <i>ModeAccessPointIdent</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 14.5: IdentCaption**

<sup>2</sup>Again, the optional aggregation is a necessary prerequisite to not break the backwards compatibility of the AUTOSAR XML schema.

<b>Class</b>	<b>ModeAccessPointIdent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
<b>Note</b>	This meta-class has been created to introduce the ability to become referenced into the meta-class Mode AccessPoint without breaking backwards compatibility.			
<b>Base</b>	<i>ARObject</i> , <i>AbstractAccessPoint</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>IdentCaption</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 14.6: ModeAccessPointIdent**

<b>Class</b>	<b>ExternalTriggeringPointIdent</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
<b>Note</b>	This meta-class has been created to introduce the ability to become referenced into the meta-class ExternalTriggeringPoint without breaking backwards compatibility.			
<b>Base</b>	<i>ARObject</i> , <i>AbstractAccessPoint</i> , <i>AtpClassifier</i> , <i>AtpFeature</i> , <i>AtpStructureElement</i> , <i>IdentCaption</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 14.7: ExternalTriggeringPointIdent**

The following (simplified) listing 14.1 sketches the usage of the meta-class *IdentCaption* for the purpose of effectively allowing references to a *ModeAccessPoint*.

**Listing 14.1: Example for the definition of an 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>

```

## 14.5 Extended Buffer Access Method

The Extended Buffer Access method enhances the support for rapid prototyping (RP) to support the bypass use case where the RTE cannot be regenerated by the bypass user.

The goal is to ensure that all `VariableDataPrototypes` that are communicated via RTE APIs are written to and read back from an `RP global buffer` that can be modified by rapid prototyping tools (RPT).

The method applies to all RTE APIs and not just those for implicit access and hence is termed the *extended* buffer access method.

Within the Extended buffer access method, a `VariableDataPrototype` can be flagged for rapid prototyping at one of three levels depending on whether post-build hooking is used. “Level 1” is intended for use by post-build hooking tools and “Level 2” and “Level 3” by non post-build hooking.

Additional RP buffers and RP flags are created when using “Level 2” and “Level 3” and the Extended Buffer Access method includes mechanisms for describing their creation and use to RP tooling.

- RP global buffer – A buffer read/written by RP. The `RP global buffer` is conceptually separated from the RTE managed buffer holding the variable data prototype value.

- RP global measurement buffer – A buffer used by RP to store the original variable data prototype value for subsequent measurement purposes before replacement by the RP generated value.
- RP enabler flag – A Boolean flag to permit run-time enabling/disabling bypass.

### 14.5.1 RP Preparation

The Extended Buffer Access method of Rapid Prototyping requires the definition of *preparation level* (see table 14.4) to enable RPT-related code generation.

The `RptProfile` of category `EXTENDED_BUFFER_ACCESS` provides the common attributes to implement the RPT support in the code.

An ECU may have to support multiple `RptProfiles` in parallel – for example, to support in one ECU the RPT tools of different suppliers.

Nevertheless, not all components might need to support all possible methods, for instance an XCP based RPT method might not be applicable for hard real-time critical functions, and therefore the `RptProfile` can be selected.

**[TPS\_SWCT\_01719] Selection of applicable `RptProfiles`** [The reference `RptContainer.explicitRptProfileSelection` provides a list of `RptProfiles` which needs to be applied when the RPT support is implemented.

If the `explicitRptProfileSelection` is not defined, all `RptProfiles` defined in the owing `RapidPrototypingScenario` are applicable.] (*RS\_SWCT\_03280*)

Class	RptProfile			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	The RptProfile describes the common properties of a Rapid Prototyping method.			
Base	<i>AObject, Identifiable, MultilanguageReferrable, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
maxServicePointId	PositiveInteger	0..1	attr	Highest service point id useable for RTE generated service points.
minServicePointId	PositiveInteger	0..1	attr	Lowest service point id useable for RTE generated service points.
servicePointSymbolPost	CIdentifier	0..1	attr	Complete symbol of the function implementing the post service point. This symbol is used for post-build hooking purposes.
servicePointSymbolPre	CIdentifier	0..1	attr	Complete symbol of the function implementing the pre service point. This symbol is used for post-build hooking purposes.
stimEnabler	<a href="#">RptEnablerImplType Enum</a>	0..1	attr	Defines if the service points support the stimulation enabler. If RptProfile.stimEnabler is "none" then no stimulation enabler is passed to the service function. Otherwise the stimulation enabler will be passed as a parameter.

**Table 14.8: RptProfile**

**[constr\_1988] Existence of attribute `RptProfile.maxServicePointId`** [For each `RptProfile`, attribute `maxServicePointId` shall exist **at the time when the RTE is generated.**]()

**[constr\_1989] Existence of attribute `RptProfile.minServicePointId`** [For each `RptProfile`, attribute `minServicePointId` shall exist **at the time when the RTE is generated.**]()

**[constr\_1990] Existence of attribute `RptProfile.servicePointSymbolPost`** [For each `RptProfile`, attribute `servicePointSymbolPost` shall exist **at the time when the RTE is generated.**]()

**[constr\_1991] Existence of attribute `RptProfile.servicePointSymbolPre`** [For each `RptProfile`, attribute `servicePointSymbolPre` shall exist **at the time when the RTE is generated.**]()

**[constr\_1992] Existence of attribute `RptProfile.stimEnabler`** [For each `RptProfile`, attribute `stimEnabler` shall exist **at the time when the RTE is generated.**]()

<b>Class</b>	<b>RptImplPolicy</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
<b>Note</b>	Describes the code preparation for rapid prototyping at data accesses.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
<code>rptEnablerImplType</code>	<code>RptEnablerImplTypeEnum</code>	0..1	attr	For Level 2 or Level3 this property determines how the RTE implements the additional "RP enabler" flag.
<code>rptPreparationLevel</code>	<code>RptPreparationEnum</code>	0..1	attr	Mandates RP preparation level for access to VariableData Prototype within generated RTE implementation.

**Table 14.9: RptImplPolicy**

**[constr\_1993] Existence of attribute `RptImplPolicy.rptEnablerImplType`** [For each `RptImplPolicy`, attribute `rptEnablerImplType` shall exist **at the time when the RTE is generated.**]()

**[constr\_1994] Existence of attribute `RptImplPolicy.rptPreparationLevel`** [For each `RptImplPolicy`, attribute `rptPreparationLevel` shall exist **at the time when the RTE is generated.**]()

<b>Enumeration</b>	<b>RptEnablerImplTypeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport::RptSupport
<b>Note</b>	Describes the required / implemented usage of enabler flags for data access in the code.
<b>Literal</b>	<b>Description</b>
none	No "RP enabler" is implemented. <b>Tags:</b> atp.EnumerationLiteralIndex=0
<code>rptEnablerRam</code>	"RP enabler" is implemented as a RAM variable <b>Tags:</b> atp.EnumerationLiteralIndex=1





Enumeration	RptEnablerImplTypeEnum
rptEnablerRamAndRom	The RTE generator implements both the RAM and ROM "RP enabler". <b>Tags:</b> atp.EnumerationLiteralIndex=3
rptEnablerRom	"RP enabler" is implemented as a calibrateable ROM variable. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 14.10: RptEnablerImplTypeEnum**

Enumeration	RptPreparationEnum
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport::RptSupport
<b>Note</b>	Determines the RP preparation level for access to VariableDataPrototypes within the generated RTE implementation.
<b>Literal</b>	<b>Description</b>
none	No RP preparation for VariableDataPrototype. <b>Tags:</b> atp.EnumerationLiteralIndex=0
rptLevel1	The RTE implementation uses an "RP global buffer" for measurement and post-build hooking purposes. <b>Tags:</b> atp.EnumerationLiteralIndex=1
rptLevel2	As rptLevel1 but the RTE implementation also uses both "RP enabler flag" to permit RP overwrite at run-time. <b>Tags:</b> atp.EnumerationLiteralIndex=2
rptLevel3	As rptLevel2 but the RTE implementation also uses "RP global measurement buffer" to record the original ECU-generated value in addition to the RP value. <b>Tags:</b> atp.EnumerationLiteralIndex=3

**Table 14.11: RptPreparationEnum**

[TPS\_SWCT\_01720] **Preparation Levels** [[RptImplPolicy.rptPreparationLevel](#)] supports three preparation levels:

- **Level 1** – If [RptImplPolicy.rptPreparationLevel](#) is set to [rptLevel1](#) then the generated RTE uses a specific memory access pattern (a write-read cycle within accessing code created by the RTE generator) suitable for access by post-build hooking tools patch writes to buffers.
- **Level 2** – If [RptImplPolicy.rptPreparationLevel](#) is set to [rptLevel2](#) then in addition to the use of an RP global buffer (as for [rptLevel1](#)) the generated code also includes an RP enabler flag that is used to make update of the RP global buffer conditional.

The RP enabler flag can be in either (calibratable) ROM or RAM based on [RptContainer.rptEnablerImplType](#).

- **Level 3** – If [RptImplPolicy.rptPreparationLevel](#) is set to [rptLevel3](#) then in addition to the requirements of [rptLevel2](#), the generated code also records the original ECU-generated value as well as the RP replacement value.

]([RS\\_SWCT\\_03280](#))



[TPS\_SWCT\_01721] References from [RptContainer](#) [If [rptImplPolicy](#) of a [RptContainer](#) is used the [RptContainer](#) can reference:

- [VariableDataPrototype](#) – the preparation level applies to a single data item.
- [ArgumentDataPrototype](#) – the preparation level applies to a single operation argument.
- [ModeDeclarationGroupPrototype](#) – the preparation level applies to a single mode.
- [operation](#) – the preparation level applies to all operation [ArgumentDataPrototype](#).
- [RunnableEntity](#) – the preparation level applies to all data items / arguments accessed by the [RunnableEntity](#).
- [SwComponentPrototype](#) – the preparation level applies to all [RunnableEntity](#)s (and hence all accessed data items and arguments) in the software component.

]([RS\\_SWCT\\_03280](#))

The generated RTE includes appropriate descriptions to enable RP tools to access the generated RP buffers and RP enabler flags.

<b>Class</b>	<b>RptSwPrototypingAccess</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport::RptSupport			
<b>Note</b>	Describes the accessibility of data and modes by the rapid prototyping tooling.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
rptHookAccess	<a href="#">RptAccessEnum</a>	0..1	attr	The related data element can be modified using a post-build hooking tool. An ENABLED VariableData Prototype is implicitly READABLE/WRITABLE.
rptReadAccess	<a href="#">RptAccessEnum</a>	0..1	attr	The related data element can be used as input for bypass functionality by RP tool. If <a href="#">rptImplPolicy</a> is not specified then RTE generation shall ensure at least suitable MC read points are created.
rptWriteAccess	<a href="#">RptAccessEnum</a>	0..1	attr	The related data element can be used as output for bypass functionality by RP tool. The data element shall be prepared to <a href="#">rptLevel2</a> and related write service points are present.

**Table 14.12: RptSwPrototypingAccess**

[constr\_1995] Existence of attribute [RptSwPrototypingAccess.rptHookAccess](#) [For each [RptSwPrototypingAccess](#), attribute [rptHookAccess](#) shall exist at the time when the RTE is generated.]()

[constr\_1996] Existence of attribute [RptSwPrototypingAccess.rptReadAccess](#) [For each [RptSwPrototypingAccess](#), attribute [rptReadAccess](#) shall exist at the time when the RTE is generated.]()

**[constr\_1997] Existence of attribute `RptSwPrototypingAccess.rptWriteAccess`** [For each `RptSwPrototypingAccess`, attribute `rptWriteAccess` shall exist at the time when the RTE is generated.]()

Enumeration	RptAccessEnum
Package	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport::RptSupport
Note	Determines the access rights to a data object with respect to rapid prototyping.
Literal	Description
enabled	The related data element is accessible by RP tool. <b>Tags:</b> atp.EnumerationLiteralIndex=0
none	The related data element is not accessible by RP tool. <b>Tags:</b> atp.EnumerationLiteralIndex=1
protected	The data element is known to the RP tool however its usage for RP can be restricted. Use case: limitation based on access rights <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table 14.13: RptAccessEnum**

## 14.5.2 Service Points

Access to the RP `global buffers` and RP `global measurement buffers` can be implemented by using a service based ECU interface in which an additional RP `Service Component`, such as an “XCP on CAN” or “XCP on Ethernet” service, is added to the ECU application.

An RP `Service Component` is an AUTOSAR or vendor specific BSW module providing an RP service, e.g. “XCP on CAN” or “XCP on Ethernet”.

It provides one or more RP `Service Function` where data is sampled and/or stimulated at an RP `Service Point`. Each RP `Service Function` call is passed an RP `Service Point ID` to identify the service point and enable different invocations to be distinguished.

The integration of the service can be performed pre-build by means of source code based integration, for example, by adding an XCP or custom BSW component, or post-build by patching the binary code of an already compiled ECU image.

In a service based scenario data is sampled and/or stimulated at RP `Service Points`. During either sampling or stimulation the data is read and/or written from the memory associated with the `VariableDataPrototype` to/from a local buffer during the execution of the RP `Service Point` and hence transferred to/from the RP tool.

Within the context of the RTE the data stimulated by the RP `Service Points` are the RP `global buffers` and RP `global measurement buffers` however any data that is measurable is potentially subject to reading.

**[TPS\_SWCT\_01722] Semantics of RP Service Point** [An `RP Service Point` is simply a call of an `RP Service Function` that is provided by the `RP Service Component`.] ([RS\\_SWCT\\_03280](#))

**[TPS\_SWCT\_01723] Semantics of RP Service Function** [The `RP Service Function` is responsible for sampling (reading) and stimulating (writing) the bypass data. The action of sampling may then trigger the RP system to perform the bypass (this may involve the communication of the sampled data to an external system for computation) ready for reading when the stimulation occurs.] ([RS\\_SWCT\\_03280](#))

Service points can be either “SWC Internal” (i.e. inserted by the SWC developer) or “RTE assigned”. SWC Internal service points are included in the SWC description (using `RapidPrototypingScenario`, see below) whereas RTE assigned are created by the RTE generator based on the specification of the SWC.

#### 14.5.2.1 Service Functions

The `RP Service Function` is responsible for sampling the required data. The sampled data is not passed as parameters – the invocation of the `RP Service Function` passes an `RP Service Point ID` as the first parameter of the `RP Service Point` which is used by the `RP Service Component` to identify the service point.

**[TPS\_SWCT\_01724] Semantics of RapidPrototypingScenario** [A `RapidPrototypingScenario` aggregates one or more `RptContainers` and one or more `RptProfiles`:

- Each `RptContainer` instance specifies, by reference, one or more element(s) applicable to the service-based access.
- Each `RptProfile` instance specifies both the name of the `RP Service Function` (attributes `servicePointSymbolPre` and `servicePointSymbolPost`) and the applicable `RP Service Point IDs` (attributes `minServicePointId` and `maxServicePointId`).

] ([RS\\_SWCT\\_03280](#))

The cross-product of the information from the `RptContainer` and `RptProfile` within a rapid prototyping scenario is used to construct the service function invocations.

Example: An `RapidPrototypingScenario` contains a single `RptContainer` that references an `RTEEvent` instance, and a single `RptProfile` with service point symbol attribute `MyServiceFunction`. As a result, the RTE generator then wraps invocations of the `RunnableEntity` started by the event with calls to service function `MyServiceFunction`.

<b>Class</b>	<b>RptExecutableEntityProperties</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
<b>Note</b>	Describes the code preparation for rapid prototyping at ExecutableEntity invocation.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
maxRptEventId	PositiveInteger	0..1	attr	Highest RPT event id usable for RTE generated service points. This attribute is relevant, if dedicated id range shall be applied to the ExecutableEntitys of a software component or specific ExecutableEntitys.
minRptEventId	PositiveInteger	0..1	attr	Lowest RPT event id usable for RTE generated service points. This attribute is relevant, if dedicated id range shall be applied to the ExecutableEntitys of a software component or specific ExecutableEntitys.
rptExecutionControl	<a href="#">RptExecutionControlEnum</a>	0..1	attr	This attribute specifies the rapid prototyping control of the executable
rptServicePoint	<a href="#">RptServicePointEnum</a>	0..1	attr	Enables generation of service points by the RTE generator.

**Table 14.14: RptExecutableEntityProperties**

**[constr\_1998] Existence of attribute [RptExecutableEntityProperties.maxRptEventId](#)** [For each [RptExecutableEntityProperties](#), attribute [maxRptEventId](#) shall exist **at the time when the RTE is generated.**]()

**[constr\_1999] Existence of attribute [RptExecutableEntityProperties.minRptEventId](#)** [For each [RptExecutableEntityProperties](#), attribute [minRptEventId](#) shall exist **at the time when the RTE is generated.**]()

**[constr\_10000] Existence of attribute [RptExecutableEntityProperties.rptExecutionControl](#)** [For each [RptExecutableEntityProperties](#), attribute [rptExecutionControl](#) shall exist **at the time when the RTE is generated.**]()

**[constr\_10001] Existence of attribute [RptExecutableEntityProperties.rptServicePoint](#)** [For each [RptExecutableEntityProperties](#), attribute [rptServicePoint](#) shall exist **at the time when the RTE is generated.**]()

<b>Enumeration</b>	<b>RptExecutionControlEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport::RptSupport
<b>Note</b>	Determines rapid prototyping preparation of an ExecutableEntity.
<b>Literal</b>	<b>Description</b>
conditional	The ExecutableEntity is only executed when the rapid prototyping disable flag is NOT set. <b>Tags:</b> atp.EnumerationLiteralIndex=0
none	The ExecutableEntity is executed without specific rapid prototyping condition. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 14.15: RptExecutionControlEnum**

<b>Enumeration</b>	<b>RptServicePointEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario
<b>Note</b>	Specifies whether the invocation of ExecutableEntitys due to activation of specific RteEvents/Bsw Events requires the insertion of Service Points.
<b>Literal</b>	<b>Description</b>
enabled	Enables generation of service points by the RTE generator. <b>Tags:</b> atp.EnumerationLiteralIndex=0
none	No Service Points are requested. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table 14.16: RptServicePointEnum**

## A 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 ([51]).

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

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

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

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

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

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

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

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

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

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

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

**AUTOSAR Partial Model** In AUTOSAR, the possible partitioning of models is marked in the meta-model by `<<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 Specification Element** An AUTOSAR Specification Element is a named element that is part of an AUTOSAR specification. Examples: requirement, constraint, specification item, class or attribute in the meta model, methodology, deliverable, methodology activity, model element, bsw module etc.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Properties are made variant by the `<<atpVariation>>`.

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

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

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

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

This is implemented using the `<<atpVariation>>`.

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

This is implemented using `EvaluatedVariantSet`.

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

This is implemented by `VariationPoint`.

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



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

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

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

This is implemented by `VariationPoint`.

## B Supported Special Use Cases

### B.1 Asymmetric Data Transformation between a Software-Component and a Complex Driver

#### B.1.1 Overview

In this scenario, a `SwComponentPrototype` typed by an `ApplicationSwComponentType` needs to communicate with a `SwComponentPrototype` typed by a `ComplexDeviceDriverSwComponentType`.

The communication itself is special insofar as it ends in an arbitrary structured data type on the side of the `ApplicationSwComponentType` and in a flat byte array on the side of the `ComplexDeviceDriverSwComponentType`.

The communication itself has a client-server nature, where the `ApplicationSwComponentType` mostly acts as the client and the `ComplexDeviceDriverSwComponentType` acts as the server.

As a consequence of this set-up, the structured data type somehow has to be serialized into a flat byte array for the call and from flat byte array to structure data type for the returning of the *out* and *inout* arguments as well as the return value.

For a justification of the use case, let's assume that the `ComplexDeviceDriverSwComponentType` implements an endpoint of a custom communication protocol that needs to interact with the "AUTOSAR world".

This is also the reason why the server implements `ClientServerOperations` with a flat byte array as the argument. The server (in this case, the `ComplexDeviceDriverSwComponentType`) itself is completely agnostic of the data, it just represents the endpoint that has to be able to deal with any given data structure.

This means that the `ComplexDeviceDriverSwComponentType` does not have to be updated and redeployed if the data structures on the side of the `ApplicationSwComponentType` undergo any changes.

For the `ApplicationSwComponentType`, on the other hand, there is little motivation to also model the respective data structures as a flat byte array. Usually, the respective data comes from a mixture of internal processing and communication with other software-components.

In other words, using a flat byte array on the side of the `ApplicationSwComponentType` would mean that the serialization has to be done anyway, potentially inside the implementation of the `ApplicationSwComponentType` itself.

Specifically, the conversion from the (structured) data type to a flat byte array and back needs to be implemented by a piece of software that is typically generated according to the structure of the data type. This software, however, is **very specific** and only fits to the corresponding data types.

One approach for data serialization in AUTOSAR is the generic concept of data transformation, in the specific case of this scenario a data transformer is needed that does a depth-first serialization over a complex data structure.

This approach is already supported in AUTOSAR by means of the so-called SOME/IP Transformer [52].

This existing concept can be taken over for the implementation of the scenario described in this chapter, albeit with some customizations. For example, the SOME/IP-specific protocol header generated by the SOME/IP Transformer is obviously not relevant for the scenario.

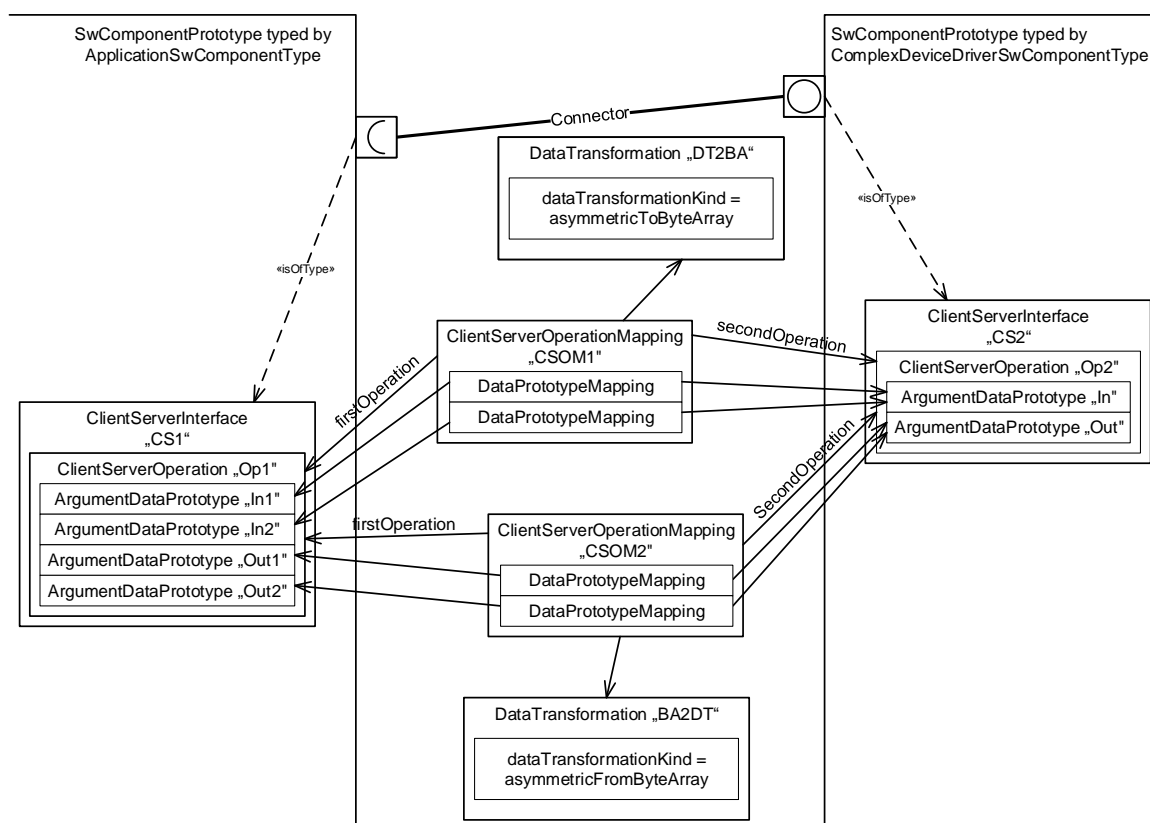
### B.1.2 Modeling Aspects

The modeling of this use case shall be explained along a simple example sketched in Figure B.1:

- It is necessary to define two individual `ClientServerOperationMappings`.
  - One of these (in this example: CSOM1) shall reference a `DataTransformation` where attribute `dataTransformationKind` is set to the value `DataTransformationKindEnum.asymmetricToByteArray`.
  - The other (in this example: CSOM2) shall reference a `DataTransformation` where attribute `dataTransformationKind` is set to the value `DataTransformationKindEnum.asymmetricFromByteArray`.
- CSOM1 shall reference the `ClientServerOperation` Op1 (on the end of the `ApplicationSwComponentType`) in the role `firstOperation`.
- CSOM1 shall reference the `ClientServerOperation` Op2 (on the end of the `ComplexDeviceDriverSwComponentType`) in the role `secondOperation`.
- CSOM2 shall reference the `ClientServerOperation` Op1 (on the end of the `ApplicationSwComponentType`) in the role `secondOperation`.
- CSOM2 shall reference the `ClientServerOperation` Op2 (on the end of the `ComplexDeviceDriverSwComponentType`) in the role `firstOperation`.
- CSOM1 shall aggregate two `DataPrototypeMappings` in the role `argumentMapping`
  - The first `DataPrototypeMapping` shall reference (in the role `firstDataPrototype`) the `ArgumentDataPrototype` named `In1` of `ClientServerOperation` OP1 and (in the role `secondDataPrototype`) the `ArgumentDataPrototype` named `In` of `ClientServerOperation` OP2.
  - The second `DataPrototypeMapping` shall reference (in the role `firstDataPrototype`) the `ArgumentDataPrototype` named `In2` of

ClientServerOperation OP1 and (in the role secondDataPrototype) the ArgumentDataPrototype named In of ClientServerOperation OP2.

- CSOM2 shall aggregate two DataPrototypeMappings in the role argumentMapping
  - The first DataPrototypeMapping shall reference (in the role firstDataPrototype) the ArgumentDataPrototype named Out1 of ClientServerOperation OP1 and (in the role secondDataPrototype) the ArgumentDataPrototype named Out of ClientServerOperation OP2.
  - The second DataPrototypeMapping shall reference (in the role firstDataPrototype) the ArgumentDataPrototype named Out2 of ClientServerOperation OP1 and (in the role secondDataPrototype) the ArgumentDataPrototype named Out of ClientServerOperation OP2.



**Figure B.1: Modeling of the data transformation use case**

Please note that in Figure B.1 the role names of references from DataPrototypeMapping have been left out of the picture for reasons of simplicity.

## C History of Constraints and Specification Items

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

### 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

Number	Heading
[ <a href="#">constr_1000</a> ]	End-to-end protection is limited to sender/receive communication
[ <a href="#">constr_1001</a> ]	Value of <a href="#">dataId</a> shall be unique
[ <a href="#">constr_1002</a> ]	End-to-end protection does not support n:1 communication
[ <a href="#">constr_1004</a> ]	Mapping of <a href="#">ApplicationDataTypes</a>
[ <a href="#">constr_1005</a> ]	Compatibility of <a href="#">ImplementationDataTypes</a> mapped to the same <a href="#">Application-DataType</a>
[ <a href="#">constr_1006</a> ]	applicable data <a href="#">categorys</a>
[ <a href="#">constr_1007</a> ]	Allowed attributes of <a href="#">SwDataDefProps</a> for <a href="#">ApplicationDataTypes</a>
[ <a href="#">constr_1008</a> ]	Applicability of <a href="#">categorys</a> STRUCTURE and ARRAY
[ <a href="#">constr_1009</a> ]	<a href="#">SwDataDefProps</a> applicable to <a href="#">ImplementationDataTypes</a>
[ <a href="#">constr_1010</a> ]	If <a href="#">nativeDeclaration</a> does not exist
[ <a href="#">constr_1011</a> ]	<a href="#">category</a> of <a href="#">SwBaseType</a>
[ <a href="#">constr_1012</a> ]	Value of <a href="#">category</a> is FIXED_LENGTH
[ <a href="#">constr_1013</a> ]	Value of <a href="#">category</a> is VARIABLE_LENGTH
[ <a href="#">constr_1014</a> ]	Supported value encodings for <a href="#">SwBaseType</a>
[ <a href="#">constr_1015</a> ]	Prioritization of <a href="#">SwDataDefProps</a>
[ <a href="#">constr_1016</a> ]	<a href="#">invalidValue</a> is restricted
[ <a href="#">constr_1017</a> ]	Supported combinations of <a href="#">swImplPolicy</a> and <a href="#">swCalibrationAccess</a>
[ <a href="#">constr_1018</a> ]	<a href="#">measurementPoint</a> shall not be referenced by a <a href="#">VariableAccess</a> aggregated by <a href="#">RunnableEntity</a> in the role <a href="#">dataReadAccess</a>
[ <a href="#">constr_1019</a> ]	Compatibility of input value and axis
[ <a href="#">constr_1020</a> ]	<a href="#">ParameterDataPrototype</a> needs to be of compatible data type as referenced in <a href="#">sharedAxisType</a>





Number	Heading
[constr_1021]	A <a href="#">CompuMethod</a> shall specify instructions for both directions
[constr_1022]	Limits shall be defined for each direction of <a href="#">CompuMethod</a>
[constr_1023]	Specification of <a href="#">Units</a> in <a href="#">CompuMethods</a>
[constr_1024]	Stepwise definition of <a href="#">CompuMethods</a>
[constr_1025]	Avoid division by zero in rational formula
[constr_1026]	Compatibility of <a href="#">Units</a>
[constr_1027]	Types for record layouts
[constr_1029]	<a href="#">ConstantSpecificationMapping</a> and <a href="#">ConstantSpecification</a>
[constr_1030]	<a href="#">ParameterSwComponentType</a> references <a href="#">ConstantSpecificationMappingSet</a>
[constr_1031]	<a href="#">NvBlockSwComponentType</a> references <a href="#">ConstantSpecificationMappingSet</a>
[constr_1032]	<a href="#">DelegationSwConnector</a> can only connect <a href="#">PortPrototypes</a> of the same kind
[constr_1033]	Communication scenarios for sender/receiver communication
[constr_1035]	Recursive definition of <a href="#">CompositionSwComponentType</a>
[constr_1036]	Connect kinds of <a href="#">PortInterfaces</a>
[constr_1037]	Client may not connect to multiple servers
[constr_1038]	Reference to <a href="#">ApplicationError</a>
[constr_1039]	Relevance of <a href="#">swImplPolicy</a>
[constr_1040]	Conversion of <a href="#">SenderReceiverInterfaces</a>
[constr_1041]	Conversion of <a href="#">ClientServerInterfaces</a>
[constr_1042]	Definition of a linear data scaling
[constr_1043]	<a href="#">PortInterface</a> vs. <a href="#">ComSpec</a>
[constr_1044]	Applicability of <a href="#">DataFilter</a>
[constr_1045]	Supported value encodings for <a href="#">SwBaseType</a> in the context of <a href="#">PortInterfaces</a>
[constr_1046]	Applicability of [constr_1045]
[constr_1047]	Compatibility of <a href="#">ApplicationPrimitiveDataTypes</a>
[constr_1048]	Compatibility of <a href="#">ApplicationRecordDataTypes</a>
[constr_1049]	Compatibility of <a href="#">ApplicationArrayDataTypes</a>
[constr_1050]	Compatibility of <a href="#">ImplementationDataTypes</a>
[constr_1051]	Compatibility of <a href="#">SwDataDefProps</a>
[constr_1052]	Compatibility of <a href="#">Units</a>
[constr_1053]	Compatibility of <a href="#">PhysicalDimensions</a>
[constr_1054]	No <a href="#">DataConstr</a> available at the provider
[constr_1055]	<a href="#">ImplementationDataType</a> has <a href="#">category</a> VALUE
[constr_1056]	<a href="#">ImplementationDataType</a> has <a href="#">category</a> TYPE_REFERENCE
[constr_1057]	<a href="#">ImplementationDataType</a> has <a href="#">category</a> DATA_REFERENCE
[constr_1058]	<a href="#">ImplementationDataType</a> has <a href="#">category</a> FUNCTION_REFERENCE
[constr_1059]	Compatibility of data types with <a href="#">category</a> VALUE
[constr_1060]	Compatibility of data types with <a href="#">category</a> ARRAY, VAL_BLK, or STRING





Number	Heading
[constr_1061]	Compatibility of data types with <code>category</code> STRUCTURE
[constr_1062]	Compatibility of data types with <code>category</code> BIT
[constr_1063]	Compatibility of data types with <code>category</code> BOOLEAN
[constr_1064]	Compatibility of data types with <code>category</code> COM_AXIS, RES_AXIS, CURVE or MAP
[constr_1066]	<code>ApplicationDataType</code> is or is not compatible to specific <code>ImplementationDataType</code>
[constr_1067]	<code>ApplicationDataType</code> is or is not compatible to specific <code>ImplementationDataType</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_1071]	compatibility of compatibility of <code>ParameterDataPrototype</code> and <code>VariableDataPrototype</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_1076]	Compatibility of <code>ArgumentDataPrototypes</code>
[constr_1077]	Compatibility of <code>ApplicationErrors</code>
[constr_1078]	Compatibility of <code>ClientServerOperations</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_1083]	Compatibility of <code>Triggers</code>
[constr_1084]	delegation of an provided outer <code>PortPrototype</code>
[constr_1085]	Compatibility in the case of a flat ECU extract
[constr_1086]	<code>SwConnector</code> between two specific <code>PortPrototypes</code>
[constr_1087]	<code>AssemblySwConnector</code> inside <code>CompositionSwComponentType</code>
[constr_1088]	<code>DelegationSwConnector</code> inside <code>CompositionSwComponentType</code>
[constr_1090]	<code>WaitPoint</code> and <code>RunnableEntity</code>
[constr_1091]	<code>RTEEvents</code> that can unblock a <code>WaitPoint</code>
[constr_1092]	<code>ParameterSwComponentType</code>
[constr_1093]	Definition of textual strings





Number	Heading
[constr_1094]	Usage of <code>symbol</code> of <code>RunnableEntity</code>
[constr_1095]	Values of <code>nDataSets</code> vs. <code>reliability</code>
[constr_1096]	<code>SwModeSwitchEvent</code> and <code>WaitPoint</code>
[constr_1097]	<code>RunnableEntity</code> that has a <code>WaitPoint</code>
[constr_1098]	Mode switch and mode disabling
[constr_1099]	Data type of inter-runnable variables
[constr_1100]	Unconnected <code>RPortPrototype</code> typed by a <code>DataInterface</code>
[constr_1101]	Mode-related communication
[constr_1102]	<code>ApplicationError</code> in the scope of one <code>SwComponentType</code>
[constr_1103]	<code>NonqueuedReceiverComSpec</code> and <code>enableUpdate</code>
[constr_1104]	Trigger sink and trigger source
[constr_1105]	Value of <code>arraySize</code>
[constr_1106]	Structure shall have at least one element
[constr_1107]	Union shall have at least one element
[constr_1108]	Value of <code>ApplicationError.errorCode</code>
[constr_1109]	Mapping of <code>SwComponentPrototypes</code> typed by a <code>SensorActuatorSwComponentType</code>
[constr_1110]	Value of <code>category</code> in <code>EndToEndDescription</code>
[constr_1111]	Constraints of <code>dataId</code> in PROFILE_01
[constr_1112]	Constraints of <code>dataIdMode</code> in PROFILE_01
[constr_1113]	Existence of attributes in PROFILE_01
[constr_1114]	Constraints of <code>crcOffset</code> in PROFILE_01
[constr_1115]	Constraints of <code>counterOffset</code> in PROFILE_01
[constr_1116]	Constraints of <code>dataLength</code> in PROFILE_01
[constr_1117]	Constraints of <code>maxDeltaCounterInit</code> in PROFILE_01
[constr_1118]	Existence of attributes in PROFILE_02
[constr_1119]	Constraints of <code>dataLength</code> in PROFILE_02
[constr_1120]	Constraints of <code>dataId</code> in PROFILE_02
[constr_1121]	Constraints of <code>maxDeltaCounterInit</code> in PROFILE_02
[constr_1122]	Existence of attributes in PROFILE_03
[constr_1123]	Constraints of <code>dataLength</code> in PROFILE_03
[constr_1124]	Constraints of <code>dataId</code> in PROFILE_03
[constr_1125]	Constraints of <code>maxDeltaCounterInit</code> in PROFILE_03
[constr_1126]	Compatibility of <code>DataConstrs</code>
[constr_2000]	Compatibility of <code>ClientServerOperations</code> triggering the same <code>RunnableEntity</code>
[constr_2001]	Initial value for a specific <code>implicitInterRunnableVariable</code> or <code>explicitInterRunnableVariable</code>
[constr_2002]	Referenced <code>VariableDataPrototype</code> from <code>AutosarVariableRef</code> of <code>VariableAccess</code> in role <code>dataReadAccess</code>
[constr_2003]	Referenced <code>VariableDataPrototype</code> from <code>AutosarVariableRef</code> of <code>VariableAccess</code> in role <code>dataWriteAccess</code>







Number	Heading
[constr_2004]	Referenced <code>VariableDataPrototype</code> from <code>AutosarVariableRef</code> of <code>VariableAccess</code> in role <code>dataSendPoint</code>
[constr_2005]	Referenced <code>VariableDataPrototype</code> from <code>AutosarVariableRef</code> of <code>VariableAccess</code> in role <code>dataReceivePointByValue</code> or <code>dataReceivePointByArgument</code>
[constr_2006]	Number of <code>AsynchronousServerCallResultPoint</code> referencing to one <code>AsynchronousServerCallPoint</code>
[constr_2007]	Consistency of <code>typeDefinition</code> attribute
[constr_2009]	Supported kinds of ports of a <code>NvBlockSwComponentType</code>
[constr_2010]	Connections between <code>SwComponentPrototypes</code> of type <code>NvBlockSwComponentType</code>
[constr_2011]	Connections between <code>SwComponentPrototypes</code> typed by <code>NvBlockSwComponentType</code> and <code>SwComponentPrototypes</code> typed by other <code>AtomicSwComponentTypes</code>
[constr_2012]	Compatibility of <code>ImplementationDataTypes</code> used for <code>ramBlock</code> and <code>romBlock</code>
[constr_2013]	Compatibility of <code>ImplementationDataTypes</code> for <code>NvBlockDataMapping</code>
[constr_2014]	Limitation of <code>RoleBasedPortAssignment.role</code> in <code>NvBlockDescriptors</code>
[constr_2015]	Limitation of <code>SwcInternalBehavior</code> of a <code>NvBlockSwComponentType</code>
[constr_2016]	Connections between <code>SwComponentPrototypes</code> of type <code>ServiceProxySwComponentType</code>
[constr_2017]	Ports of <code>ServiceProxySwComponentTypes</code>
[constr_2018]	Supported remote communication of a <code>ServiceProxySwComponentType</code>
[constr_2019]	<code>ServiceSwComponentType</code> shall have service ports only
[constr_2020]	<code>dataReadAccess</code> can not be used for queued communication
[constr_2021]	<code>WaitPoint</code> referencing a <code>DataReceivedEvent</code> can not be used for non-queued communication
[constr_2022]	Mutually exclusive use of <code>SynchronousServerCallPoints</code> and <code>AsynchronousServerCallPoints</code>
[constr_2023]	Consistency of <code>timeout</code> values
[constr_2024]	<code>enableTakeAddress</code> is restricted to single instantiation
[constr_2025]	Uniqueness of <code>symbol</code> attributes
[constr_2026]	Referenced <code>VariableDataPrototype</code> from <code>AutosarVariableRef</code> of <code>VariableAccess</code> in role <code>writtenLocalVariable</code> and <code>readLocalVariable</code>
[constr_2027]	<code>SwcServiceDependency</code> shall be defined for service ports only
[constr_2028]	<code>staticMemory</code> is restricted to single instantiation
[constr_2029]	<code>shortName</code> of <code>constantMemory</code> and <code>staticMemory</code>
[constr_2030]	<code>AsynchronousServerCallResultPoint</code> combined with <code>WaitPoint</code> shall belong to the same <code>RunnableEntity</code>
[constr_2031]	Period of <code>TimingEvent</code> shall be greater than 0
[constr_2032]	<code>transmissionAcknowledge</code> requires a <code>DataSendCompletedEvent</code>
[constr_2033]	Timeout of <code>DataSendCompletedEvent</code>
[constr_2500]	<code>PortInterfaces</code> shall be of same kind





Number	Heading
[constr_2526]	<a href="#">PortInterface</a> s need to be compatible to the blueprints
[constr_2527]	Blueprints shall live in package of a proper category
[constr_2528]	<a href="#">PortPrototype</a> s shall not refer to blueprints of <a href="#">PortInterfaces</a>
[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 <a href="#">ModeRequestTypeMap</a>
[constr_4002]	Unambiguous mapping of modes to data types
[constr_4003]	Semantics of <a href="#">SwcModeSwitchEvent</a>
[constr_4004]	Context of <a href="#">SenderReceiverAnnotation</a>
[constr_4005]	Context of <a href="#">ClientServerAnnotation</a>
[constr_4006]	Context of <a href="#">ParameterPortAnnotation</a>
[constr_4007]	Context of <a href="#">ModePortAnnotation</a>
[constr_4008]	Context of <a href="#">TriggerPortAnnotation</a>
[constr_4009]	Context of <a href="#">NvDataPortAnnotation</a>
[constr_4010]	Context of <a href="#">DelegatedPortAnnotation</a>
[constr_4011]	ComSpec and <a href="#">ModeSwitchedAckEvent</a>
[constr_4012]	Timeout of <a href="#">ModeSwitchedAckEvent</a>
[constr_4035]	<a href="#">ValueSpecification</a> shall fit into data type

**Table C.1: Added Constraints in R4.0.1**

### C.1.3 Deleted Constraints in R4.0.1

N/A

## C.2 Constraint History of this Document according to AUTOSAR R4.0.2

### C.2.1 Changed Constraints in R4.0.2

Number	Heading
[constr_1007]	Allowed attributes of <a href="#">SwDataDefProps</a> for <a href="#">ApplicationDataTypes</a>
[constr_1061]	Compatibility of data types with <a href="#">category</a> STRUCTURE
[constr_2001]	Initial value for a specific <a href="#">implicitInterRunnableVariable</a> or <a href="#">explicitInterRunnableVariable</a>

**Table C.2: Changed Constraints in R4.0.2**

## C.2.2 Added Constraints in R4.0.2

Number	Heading
[constr_1127]	<code>ServiceSwComponentType</code> shall not have <code>ServiceNeeds</code>
[constr_1128]	Queue length of <code>ClientServerOperations</code> associated with the same <code>RunnableEntity</code>
[constr_1129]	<code>swImplPolicy</code> and <code>NonqueuedReceiverComSpec</code>
[constr_1130]	<code>swImplPolicy</code> and <code>NonqueuedReceiverComSpec</code>
[constr_1131]	<code>swImplPolicy</code> and <code>NonqueuedSenderComSpec</code>
[constr_1132]	<code>swImplPolicy</code> and <code>NonqueuedSenderComSpec</code>
[constr_1133]	Identical <code>CompuScale</code> Symbolic Names shall have the same range
[constr_1134]	Allowed structure of <code>TEXTTABLE</code>
[constr_1135]	Limit of <code>vt</code> in <code>BITFIELD_TEXTTABLE</code>
[constr_1136]	Compatibility of introduction of blueprint and blueprinted element
[constr_1137]	Applicability of <code>ParameterInterface</code>
[constr_1138]	<code>assignedPort</code> and <code>DiagEventDebounceMonitorInternal</code>
[constr_1139]	<code>assignedPort</code> of <code>DiagEventDebounceMonitorInternal</code> shall refer to an <code>RPortPrototype</code>
[constr_2034]	<code>SwAddrMethod</code> referenced by <code>RunnableEntity</code> s or <code>BswSchedulableEntity</code> s
[constr_2035]	<code>swImplPolicy</code> for <code>VariableDataPrototype</code> in <code>SenderReceiverInterface</code>
[constr_2036]	<code>swImplPolicy</code> for <code>VariableDataPrototype</code> in <code>NvDataInterface</code>
[constr_2037]	<code>swImplPolicy</code> for <code>VariableDataPrototype</code> in the role <code>ramBlock</code>
[constr_2038]	<code>swImplPolicy</code> for <code>VariableDataPrototype</code> in the role <code>implicitInter-RunnableVariable</code>
[constr_2039]	<code>swImplPolicy</code> for <code>VariableDataPrototype</code> in the role <code>explicitInter-RunnableVariable</code>
[constr_2040]	<code>swImplPolicy</code> for <code>VariableDataPrototype</code> in the role <code>arTypedPerInstance-Memory</code>
[constr_2041]	<code>swImplPolicy</code> for <code>VariableDataPrototype</code> in the role <code>staticMemory</code>
[constr_2042]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in <code>ParameterInterface</code>
[constr_2043]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in the role <code>staticMemory</code>
[constr_2044]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in the role <code>sharedParameter</code>
[constr_2045]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in the role <code>perInstanceParameter</code>
[constr_2046]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in the role <code>constantMemory</code>
[constr_2047]	<code>swImplPolicy</code> for <code>ArgumentDataPrototype</code>
[constr_2048]	<code>swImplPolicy</code> for <code>SwServiceArg</code>
[constr_2535]	Target of an <code>autosarParameter</code> in <code>AutosarParameterRef</code> shall refer to a parameter
[constr_2536]	Target of an <code>autosarVariable</code> in <code>AutosarVariableRef</code> 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

Number	Heading
[constr_1006]	applicable data <a href="#">categorys</a>
[constr_1009]	<a href="#">SwDataDefProps</a> applicable to <a href="#">ImplementationDataTypes</a>
[constr_1014]	Supported value encodings for <a href="#">SwBaseType</a>
[constr_1015]	Prioritization of <a href="#">SwDataDefProps</a>
[constr_1043]	<a href="#">PortInterface</a> vs. <a href="#">ComSpec</a>
[constr_1051]	Compatibility of <a href="#">SwDataDefProps</a>
[constr_1053]	Compatibility of <a href="#">PhysicalDimensions</a>
[constr_1063]	Compatibility of data types with <a href="#">category</a> BOOLEAN
[constr_1110]	Value of <a href="#">category</a> in <a href="#">EndToEndDescription</a>
[constr_1113]	Existence of attributes in PROFILE_01
[constr_1118]	Existence of attributes in PROFILE_02
[constr_1134]	Allowed structure of <a href="#">TEXTTABLE</a>
[constr_2000]	Compatibility of <a href="#">ClientServerOperations</a> triggering the same <a href="#">RunnableEntity</a>
[constr_2027]	<a href="#">SwcServiceDependency</a> 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

Number	Heading
[constr_1140]	Combination of <a href="#">invalidValue</a> with the attribute <a href="#">handleInvalid</a>
[constr_1141]	Applicability of the <a href="#">scope</a> attribute
[constr_1142]	<a href="#">category</a> of <a href="#">CompuMethod</a> shall not be extended
[constr_1143]	<a href="#">category</a> of <a href="#">AutosarDataType</a> shall not be extended
[constr_1144]	<a href="#">SensorActuatorSwComponentType</a> , <a href="#">EcuAbstractionSwComponentType</a> , and <a href="#">ComplexDeviceDriverSwComponentType</a> may only reference a <a href="#">HwType</a>





Number	Heading
[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>categorys</code> 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 <code>unit</code>
[constr_1176]	Compatibility of <code>CompuScales</code> of <code>category</code> <code>LINEAR</code> and <code>RAT_FUNC</code>



△

Number	Heading
[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

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>SwcServiceDependencies</code> 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>categorys</code> 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 <code>ApplicationCompositeDataType</code> and nested <code>ImplementationDataType</code>
[TPS_SWCT_01025]	The role of <code>PortPrototypes</code> in the AUTOSAR architecture
[TPS_SWCT_01026]	The role of <code>PortInterfaces</code> in the AUTOSAR architecture
[TPS_SWCT_01027]	Different flavors of <code>PortInterfaces</code>
[TPS_SWCT_01028]	<code>AtomicSwComponentType</code> implements a Diagnostic Monitor





Number	Heading
[TPS_SWCT_01029]	<code>AtomicSwComponentType</code> implements a Diagnostic Monitor
[TPS_SWCT_01030]	<code>RunnableEntity</code>
[TPS_SWCT_01031]	<code>ExclusiveArea</code>
[TPS_SWCT_01032]	<code>CompositionSwComponentType</code>
[TPS_SWCT_01033]	Nested definition of <code>CompositionSwComponentTypes</code>
[TPS_SWCT_01034]	<code>CompositionSwComponentTypes</code> do not have any binary footprint
[TPS_SWCT_01035]	<code>CompositionSwComponentType</code> aggregates <code>SwComponentPrototypes</code>
[TPS_SWCT_01036]	<code>SwComponentPrototype</code> implements a specific role
[TPS_SWCT_01037]	arbitrary numbers of <code>SwComponentPrototypes</code> 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]	<code>SwConnector</code> exists depending on a <code>PostBuild</code> condition
[TPS_SWCT_01041]	API functions of not existing <code>SwConnector</code> 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]	<code>ApplicationSwComponentTypes</code> are independent from actual ECU Hardware
[TPS_SWCT_01044]	<code>ServiceNeeds</code>
[TPS_SWCT_01045]	Actual values of ECU configuration parameters fulfill the requirements given by the <code>ServiceNeeds</code>
[TPS_SWCT_01046]	<code>ServiceNeeds</code> are defined in the scope of the <code>SwcInternalBehavior</code>
[TPS_SWCT_01047]	Reference from the software representation of a sensor/actuator to the actual hardware element
[TPS_SWCT_01048]	<code>SensorActuatorSwComponentType</code> may use the I/O hardware abstraction directly
[TPS_SWCT_01049]	Two ways to use the <code>ExclusiveAreas</code>
[TPS_SWCT_01050]	<code>RunnableEntity</code> always runs inside an <code>ExclusiveArea</code>
[TPS_SWCT_01051]	<code>RunnableEntity</code> explicitly enters and leaves a specific <code>ExclusiveArea</code>
[TPS_SWCT_01052]	Inter-runnable variable
[TPS_SWCT_01053]	Relationship of interchanged data with <code>RunnableEntities</code>
[TPS_SWCT_01054]	Semantics of the <code>explicitInterRunnableVariable</code>
[TPS_SWCT_01055]	Semantics of <code>implicitInterRunnableVariable</code>
[TPS_SWCT_01056]	Physical dimension
[TPS_SWCT_01057]	Unit references one physical dimension
[TPS_SWCT_01058]	<code>UnitGroup</code>
[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







Number	Heading
[TPS_SWCT_01063]	<a href="#">PortGroup</a>
[TPS_SWCT_01064]	<a href="#">PortGroups</a> have to be defined on the VFB level
[TPS_SWCT_01065]	<a href="#">PortPrototype</a> may belong to more than one <a href="#">PortGroups</a>
[TPS_SWCT_01066]	<a href="#">PortGroups</a> can be associated with certain <a href="#">ServiceNeeds</a>
[TPS_SWCT_01067]	Initial mode
[TPS_SWCT_01068]	<a href="#">Units</a> can be grouped with the help of <a href="#">UnitGroup</a>
[TPS_SWCT_01069]	<a href="#">DataInterface</a> is defined as abstract base class
[TPS_SWCT_01070]	<a href="#">PortInterface</a> acts as a <i>type</i> for a <a href="#">PortPrototype</a>
[TPS_SWCT_01071]	<a href="#">ModeDeclaration</a>
[TPS_SWCT_01072]	<a href="#">ApplicationDataType</a> and <a href="#">ImplementationDataType</a>
[TPS_SWCT_01073]	Composite <a href="#">ApplicationDataType</a>
[TPS_SWCT_01074]	Composite <a href="#">ImplementationDataType</a>
[TPS_SWCT_01075]	<a href="#">SwcInternalBehavior</a>
[TPS_SWCT_01076]	Number of elements of a specific <a href="#">ApplicationArrayDataType</a> might vary at run-time
[TPS_SWCT_01077]	Configure the response to mode changes
[TPS_SWCT_01078]	Configurable array size
[TPS_SWCT_01079]	<a href="#">SwConnector</a>
[TPS_SWCT_01080]	Delegation ports
[TPS_SWCT_01081]	Implications of being a delegation port
[TPS_SWCT_01082]	<a href="#">AssemblySwConnector</a>
[TPS_SWCT_01083]	<a href="#">DelegationSwConnector</a>
[TPS_SWCT_01084]	Outer <a href="#">PortPrototype</a> is referenced by multiple <a href="#">DelegationSwConnectors</a>
[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 <a href="#">CompositionSwComponentTypes</a>
[TPS_SWCT_01089]	end-to-end communication protection
[TPS_SWCT_01090]	<a href="#">EndToEndProtection</a>
[TPS_SWCT_01091]	Two cases for end-to-end protection
[TPS_SWCT_01092]	<a href="#">EndToEndProtectionSet</a>
[TPS_SWCT_01093]	Definition of end-to-end protection is splittable
[TPS_SWCT_01094]	<i>category</i> of <a href="#">EndToEndDescription</a>
[TPS_SWCT_01095]	<i>category</i> set to NONE
[TPS_SWCT_01096]	<a href="#">PortGroup</a>
[TPS_SWCT_01097]	<a href="#">CompositionSwComponentType</a> cannot have <a href="#">RunnableEntities</a>
[TPS_SWCT_01098]	Only <a href="#">AtomicSwComponentType</a> can have <a href="#">RunnableEntities</a>
[TPS_SWCT_01099]	<a href="#">PortInterfaceMapping</a>





Number	Heading
[TPS_SWCT_01100]	Precedence of <a href="#">PortInterfaceMapping</a>
[TPS_SWCT_01101]	Unmapped elements of <a href="#">PortInterfaces</a>
[TPS_SWCT_01102]	<a href="#">VariableAndParameterInterfaceMapping</a>
[TPS_SWCT_01103]	Mapping between different kinds of <a href="#">PortInterfaces</a>
[TPS_SWCT_01104]	Possible mappings are restricted by the <a href="#">swImplPolicy</a>
[TPS_SWCT_01105]	<a href="#">ClientServerInterfaceMapping</a>
[TPS_SWCT_01106]	<a href="#">ClientServerOperation</a>
[TPS_SWCT_01107]	<a href="#">swMinAxisPoints</a> and <a href="#">swMaxAxisPoints</a> represent variation points
[TPS_SWCT_01108]	Added value of an <a href="#">AtomicSwComponentType</a>
[TPS_SWCT_01109]	Adding the <a href="#">SwcInternalBehavior</a> in a later process step
[TPS_SWCT_01110]	Symbolic name of a software-component
[TPS_SWCT_01111]	<a href="#">PortPrototypes</a> need an additional model artifact, the <a href="#">PortInterface</a>
[TPS_SWCT_01112]	<a href="#">PortPrototypes</a> are either <i>require-</i> or <i>provide-</i> ports.
[TPS_SWCT_01113]	Connecting two <a href="#">PortPrototypes</a>
[TPS_SWCT_01114]	<a href="#">SenderReceiverInterface</a>
[TPS_SWCT_01115]	<a href="#">invalidationPolicy</a>
[TPS_SWCT_01116]	<a href="#">swImplPolicy</a>
[TPS_SWCT_01117]	Communication patterns for sender-receiver communication
[TPS_SWCT_01118]	<a href="#">ClientServerInterface</a>
[TPS_SWCT_01119]	Direction of <a href="#">ArgumentDataPrototypes</a>
[TPS_SWCT_01120]	Client needs to provide <a href="#">ArgumentDataPrototypes</a>
[TPS_SWCT_01121]	Pass correct data type
[TPS_SWCT_01122]	Synchronous call of <a href="#">ClientServerOperation</a>
[TPS_SWCT_01123]	No default values for <a href="#">ArgumentDataPrototypes</a>
[TPS_SWCT_01124]	Definition of <a href="#">ArgumentDataPrototypes</a> within the context of a <a href="#">ClientServerOperation</a> is ordered
[TPS_SWCT_01125]	<a href="#">serverArgumentImplPolicy</a>
[TPS_SWCT_01126]	Access to partial networking via BswM
[TPS_SWCT_01127]	Byte array with variable size
[TPS_SWCT_01128]	<a href="#">SwRecordLayout</a> needed
[TPS_SWCT_01129]	Express diagnostic capabilities
[TPS_SWCT_01130]	Measurement and calibration access to model elements is defined by <a href="#">swCalibrationAccess</a>
[TPS_SWCT_01131]	<a href="#">AtomicSwComponentType</a> accepts a request to restart an entire function
[TPS_SWCT_01132]	<a href="#">AtomicSwComponentType</a> provides information about operating cycles
[TPS_SWCT_01133]	<a href="#">AtomicSwComponentType</a> provides information about aging cycles
[TPS_SWCT_01134]	<a href="#">AtomicSwComponentType</a> enables storage of DTCs in general
[TPS_SWCT_01135]	<a href="#">AtomicSwComponentType</a> enables storage of subsequent DTCs
[TPS_SWCT_01136]	<a href="#">AtomicSwComponentType</a> retrieves information from the fault storage





Number	Heading
[TPS_SWCT_01137]	Dem provides information that the fault storage overflows
[TPS_SWCT_01138]	<code>AtomicSwComponentType</code> suppresses the storage of DTCs within the Dem
[TPS_SWCT_01139]	<code>AtomicSwComponentType</code> informs the Dem that the PTO is active
[TPS_SWCT_01140]	<code>AtomicSwComponentType</code> needs information about specific DTC without being a diagnostic monitor
[TPS_SWCT_01141]	<code>AtomicSwComponentType</code> may have <code>RPortPrototypes</code> typed by an <code>NvDataInterface</code>
[TPS_SWCT_01142]	non-volatile data are provided by a specialized <code>AtomicSwComponentType</code>
[TPS_SWCT_01143]	Non-volatile data represented by an <code>NvBlockComponent</code> can be read and written
[TPS_SWCT_01144]	<code>NvBlockDescriptor</code> specifies the properties of exactly one <code>NvBlock</code>
[TPS_SWCT_01145]	<code>ramBlock</code> and the <code>romBlock</code> are described by a <code>VariableDataPrototype</code> and a <code>ParameterDataPrototype</code>
[TPS_SWCT_01146]	<code>romBlock</code> is optional
[TPS_SWCT_01147]	No <code>romBlock</code> is configured
[TPS_SWCT_01148]	<code>NvBlockDataMapping</code>
[TPS_SWCT_01149]	<code>RoleBasedPortAssignment</code> of <code>NvBlockDescriptor</code>
[TPS_SWCT_01150]	<code>InternalBehavior</code> of a <code>NvBlockSwComponentType</code>
[TPS_SWCT_01151]	<code>RunnableEntity</code> s do not have further attributes
[TPS_SWCT_01152]	<code>InternalBehavior</code> does not have further attributes
[TPS_SWCT_01153]	<code>IncludedModeDeclarationGroupSet</code>
[TPS_SWCT_01154]	Attribute <code>prefix</code> of <code>IncludedModeDeclarationGroupSet</code>
[TPS_SWCT_01155]	<code>IncludedDataTypeSet</code>
[TPS_SWCT_01156]	Required if the <code>AutosarDataType</code> is not used for any <code>DataPrototype</code>
[TPS_SWCT_01157]	Attribute <code>literalPrefix</code> of <code>IncludedDataTypeSet</code>
[TPS_SWCT_01158]	Three cases for <code>PortInterfaceMapping</code>
[TPS_SWCT_01159]	Mapping is described separately from the <code>SwConnector</code> as reusable <code>ARElement</code>
[TPS_SWCT_01160]	<code>ModeInterfaceMapping</code>
[TPS_SWCT_01161]	<code>TriggerInterfaceMapping</code>
[TPS_SWCT_01162]	Conditional existence of <code>TextTableMapping</code>
[TPS_SWCT_01163]	Conversion from <code>firstValue</code> to <code>secondValue</code>
[TPS_SWCT_01164]	Conversion from <code>secondValue</code> to <code>firstValue</code>
[TPS_SWCT_01165]	Invertible mapping
[TPS_SWCT_01166]	Non-invertible mapping
[TPS_SWCT_01167]	Validity of <code>ModeInterfaceMapping</code>
[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





Number	Heading
[TPS_SWCT_01172]	Requesting and releasing partial networks
[TPS_SWCT_01173]	Control port shall <b>not</b> become a part of the <a href="#">PortGroup</a>
[TPS_SWCT_01174]	Status port shall <b>not</b> become a member of the <a href="#">PortGroup</a>
[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 <a href="#">ValueSpecification</a>
[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 <a href="#">SwSystemconst</a>
[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 <a href="#">Autosar-DataType</a>
[TPS_SWCT_01184]	<a href="#">ApplicationPrimitiveDataTypes</a> with <code>category</code> VALUE
[TPS_SWCT_01185]	<code>initValues</code> for Compound Primitive Data Types
[TPS_SWCT_01186]	<a href="#">ConstantSpecificationMapping</a>
[TPS_SWCT_01187]	<a href="#">ConstantSpecificationMappingSet</a> referenced by the <a href="#">InternalBehavior</a>
[TPS_SWCT_01188]	Definition of calibration data sets through RTE-generator and compiler
[TPS_SWCT_01189]	<a href="#">DataTypeMap</a>
[TPS_SWCT_01190]	<a href="#">ModeRequestTypeMap</a>
[TPS_SWCT_01191]	mapped <a href="#">ApplicationDataType</a> and <a href="#">ImplementationDataType</a> shall be compatible
[TPS_SWCT_01192]	Meta-classes that have an association to a <a href="#">DataTypeMappingSet</a>
[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 <a href="#">ImplementationDataType</a>
[TPS_SWCT_01195]	Mapping of composite element to primitive <a href="#">DataPrototype</a>
[TPS_SWCT_01196]	Semantics of an external trigger event communication
[TPS_SWCT_01197]	<a href="#">TriggerInterface</a>
[TPS_SWCT_01198]	Period for periodic triggering
[TPS_SWCT_01199]	Queued processing of <a href="#">Triggers</a>
[TPS_SWCT_01200]	<a href="#">ModeDeclarationGroupPrototype</a> per <a href="#">ModeSwitchInterface</a>
[TPS_SWCT_01201]	<a href="#">CompositionSwComponentType</a> requires and provides the modes that are required or provided by its contained <a href="#">SwComponentPrototypes</a>
[TPS_SWCT_01202]	<a href="#">ApplicationDataType</a> defines a subset of the values used in the <a href="#">ModeDeclarationGroup</a>
[TPS_SWCT_01203]	<a href="#">PortPrototype</a> may own port annotations
[TPS_SWCT_01204]	<a href="#">GeneralAnnotation</a>
[TPS_SWCT_01205]	Typical annotations for sender/receiver communication





Number	Heading
[TPS_SWCT_01206]	Min and Max annotations are valid for a certain amount of time
[TPS_SWCT_01207]	<code>VariableDataPrototypes</code> use the same application-level <code>SenderReceiverAnnotation</code>
[TPS_SWCT_01208]	Grouping for <code>SenderReceiverAnnotation</code>
[TPS_SWCT_01209]	<code>ClientServerAnnotation</code>
[TPS_SWCT_01210]	<code>IoHwAbstractionServerAnnotation</code>
[TPS_SWCT_01211]	Assign several annotations to <code>ArgumentDataPrototype</code>
[TPS_SWCT_01212]	<code>ParameterPortAnnotation</code>
[TPS_SWCT_01213]	<code>ModePortAnnotation</code>
[TPS_SWCT_01214]	<code>TriggerPortAnnotation</code>
[TPS_SWCT_01215]	<code>NvDataPortAnnotation</code>
[TPS_SWCT_01216]	<code>DelegatedPortAnnotation</code>
[TPS_SWCT_01217]	Semantics of <code>DelegatedPortAnnotation.signalFan</code>
[TPS_SWCT_01218]	Big picture of <code>ComSpec</code>
[TPS_SWCT_01219]	<code>ComSpec</code> for queued and non-queued sender-receiver communication
[TPS_SWCT_01220]	<code>initValue</code> defines an initial value that shall be taken if the corresponding <code>dataElement</code> has not yet been received
[TPS_SWCT_01221]	<code>DataFilter</code>
[TPS_SWCT_01222]	Applicability of <code>DataFilter</code>
[TPS_SWCT_01223]	<code>networkRepresentation</code> defines how a specific <code>dataElement</code> is represented on a communication bus
[TPS_SWCT_01224]	<code>CompuMethods</code> of <code>dataElement</code> and the <code>networkRepresentation</code> are used for conversion purposes
[TPS_SWCT_01225]	<code>RunnableEntity</code> implements the functionality of two or more <code>ClientServerOperations</code>
[TPS_SWCT_01226]	<code>initValue</code> on the level of a <code>ComSpec</code> is relevant for connections to the corresponding <code>PortPrototype</code>
[TPS_SWCT_01227]	Unconnected <code>RPortPrototype</code> typed by <code>NvDataInterface</code>
[TPS_SWCT_01228]	<code>NvProvideComSpec</code>
[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]	<code>SwDataDefProps</code>
[TPS_SWCT_01238]	Attribute <code>category</code> used in the context of <code>AutosarDataType</code>





Number	Heading
[TPS_SWCT_01239]	default value for attribute <code>category</code> used in the context of <code>AutosarDataType</code>
[TPS_SWCT_01240]	Subclasses of <code>ApplicationDataType</code>
[TPS_SWCT_01241]	Applicable <code>category</code> s for subclasses <code>ApplicationDataType</code>
[TPS_SWCT_01242]	<code>category</code> 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]	<code>SwDataDefProps</code> control the structure of calibration parameters
[TPS_SWCT_01246]	<code>SwRecordLayout</code> may be required for A2L generation
[TPS_SWCT_01247]	<code>ApplicationArrayDataType</code> and <code>ApplicationRecordDataType</code>
[TPS_SWCT_01248]	Nested definition of <code>ImplementationDataType</code>
[TPS_SWCT_01249]	<code>ApplicationRecordDataType</code>
[TPS_SWCT_01250]	<code>ImplementationDataType</code> has been introduced to optimize the formal support for data type handling on the implementation level
[TPS_SWCT_01251]	Limited set of values for <code>category</code> are applicable for <code>Implementation-DataType</code>
[TPS_SWCT_01252]	<code>ImplementationDataType</code> can express concepts not available on application level
[TPS_SWCT_01253]	Rules apply for the usage of the attribute <code>ImplementationDataType.type- Emitter</code>
[TPS_SWCT_01254]	<code>ImplementationDataType</code> 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]	<code>ImplementationDataType</code> or the aggregated <code>Implementation-DataTypeElements</code> 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]	<code>SwBaseType</code>
[TPS_SWCT_01261]	Use case for <code>SwBaseType</code>
[TPS_SWCT_01262]	<code>memAlignment</code> and <code>byteOrder</code> are platform specific
[TPS_SWCT_01263]	Further use cases for <code>SwBaseType</code>
[TPS_SWCT_01264]	Data prototypes implement a role of a data type
[TPS_SWCT_01265]	<code>DataPrototype</code> aggregates an own set of <code>SwDataDefProps</code>
[TPS_SWCT_01266]	Three non-abstract classes derived from <code>AutosarDataPrototype</code>
[TPS_SWCT_01267]	<code>DataPrototype</code> can be aggregated in different roles
[TPS_SWCT_01268]	Definition of <code>initValue</code> for a <code>VariableDataPrototype</code> or a <code>Parameter-DataPrototype</code>
[TPS_SWCT_01269]	In <code>PortInterfaces</code> , initial values defined for <code>DataPrototypes</code> are ignored
[TPS_SWCT_01270]	<code>AutosarVariableRef</code>
[TPS_SWCT_01271]	<code>AutosarParameterRef</code>
[TPS_SWCT_01272]	Semantics of <code>swComparisonVariable</code>





Number	Heading
[TPS_SWCT_01273]	Precedence rules for the application of <code>SwDataDefProps</code>
[TPS_SWCT_01274]	<code>SwDataDefProps</code> used to support calibration and measurement
[TPS_SWCT_01275]	values of the attribute <code>swImplPolicy</code> 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]	<code>CompuMethods</code> 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]	<code>CompuMethod</code> applied to values outside of its limits
[TPS_SWCT_01281]	<code>Unit</code> associated with a <code>PhysicalDimension</code>
[TPS_SWCT_01283]	Rational function
[TPS_SWCT_01284]	<code>CompuScale</code> might require a representation in the generated RTE C code
[TPS_SWCT_01285]	Physical dimension
[TPS_SWCT_01286]	<code>DataConstr</code>
[TPS_SWCT_01287]	Standard limits and extended limits in the ASAM-MCD2 (ASAP2) specification
[TPS_SWCT_01288]	Interpretation of <code>PhysConstrs</code> and <code>InternalConstrs</code> by tools
[TPS_SWCT_01289]	Semantics of <code>Limit</code>
[TPS_SWCT_01290]	<code>SwAddrMethod</code>
[TPS_SWCT_01291]	Association of <code>MemorySection</code> with <code>SwAddrMethod</code>
[TPS_SWCT_01292]	Usage of <code>SwAddrMethod</code> in the context of a <code>DataPrototype</code>
[TPS_SWCT_01293]	RTE Generator has to derive the Memory Allocation Keyword
[TPS_SWCT_01294]	Missing <code>SwDataDefProps.swAddrMethod</code>
[TPS_SWCT_01295]	<code>SwRecordLayout</code>
[TPS_SWCT_01296]	Different approaches of ASAM MCD-2MC and AUTOSAR with respect to <code>SwRecordLayout</code>
[TPS_SWCT_01297]	Compliance of <code>ApplicationDataTypes</code> or <code>ImplementationDataTypes</code> to <code>swDataDefProps</code>
[TPS_SWCT_01298]	Computing <code>SwRecordLayout</code> from <code>ImplementationDataTypes</code> is not possible
[TPS_SWCT_01299]	Relation of <code>swRecordLayoutGroup</code> to <code>subElement</code>
[TPS_SWCT_01300]	Relationship between record layouts and interpolation routines
[TPS_SWCT_01301]	Importance of initial values
[TPS_SWCT_01302]	Semantics of <code>minimumStartInterval</code>
[TPS_SWCT_01303]	<code>symbol</code> attribute describes the <code>RunnableEntity</code> 's entry point
[TPS_SWCT_01304]	Cat. 1A and 1B <code>RunnableEntity</code> s will eventually terminate
[TPS_SWCT_01305]	<code>RunnableEntity</code> 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 <code>RunnableEntity</code>
[TPS_SWCT_01307]	<code>supportsMultipleInstantiation</code> vs. <code>canBeInvokedConcurrently</code>





Number	Heading
[TPS_SWCT_01308]	Combination of <code>supportsMultipleInstantiation=false</code> and <code>canBeInvokedConcurrently=false</code>
[TPS_SWCT_01309]	signature of a <code>RunnableEntity</code> depends on the connected <code>RTEEvent</code>
[TPS_SWCT_01310]	Categories of <code>RunnableEntities</code>
[TPS_SWCT_01311]	Name of an operation argument
[TPS_SWCT_01312]	<code>RunnableEntity</code> has a mapping to <code>BswModuleEntry</code>
[TPS_SWCT_01313]	Conditions for a transition from <code>suspended</code> to to be started
[TPS_SWCT_01314]	<code>RTEEvent</code>
[TPS_SWCT_01315]	Interaction of <code>RunnableEntity</code> with <code>RTEEvent</code>
[TPS_SWCT_01316]	Abstract base class <code>RTEEvent</code>
[TPS_SWCT_01317]	RTE triggers <code>RunnableEntity</code> in response to occurring <code>RTEEvent</code>
[TPS_SWCT_01318]	<code>RunnableEntity</code> and <code>WaitPoint</code>
[TPS_SWCT_01319]	<code>RTEEvent</code> can be used to trigger <code>WaitPoints</code> in different <code>RunnableEntities</code>
[TPS_SWCT_01320]	<code>RunnableEntities</code> of category 2
[TPS_SWCT_01321]	Communication among <code>RunnableEntities</code>
[TPS_SWCT_01322]	Interaction patterns for the application of the sender-receiver paradigm
[TPS_SWCT_01323]	Read and write access to a <code>dataElement</code>
[TPS_SWCT_01324]	Mode switches need to be completed in finite time
[TPS_SWCT_01325]	Read and write access is only applicable for <code>RunnableEntities</code> 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 <code>scope</code>
[TPS_SWCT_01329]	Access to specific data is implemented by means of aggregating the meta-class <code>VariableAccess</code> in specific roles
[TPS_SWCT_01330]	<code>RunnableEntity</code> can also have <code>dataSendPoints</code>
[TPS_SWCT_01331]	<code>dataWriteAccess</code> vs. <code>dataSendPoint</code>
[TPS_SWCT_01332]	<code>dataReceivePointByValue</code> vs. <code>dataReceivePointByArgument</code>
[TPS_SWCT_01333]	<code>dataReceivePointByValue/dataReceivePointByArgument</code> vs. <code>dataReadAccess</code>
[TPS_SWCT_01334]	<code>RunnableEntities</code> of category 1 may have <code>dataReceivePointByValues/dataReceivePointByArguments</code>
[TPS_SWCT_01335]	Combine <code>dataReceivePointByValue</code> or <code>dataReceivePointByArgument</code> with a <code>WaitPoint</code>
[TPS_SWCT_01336]	<code>dataSendPoint</code> also allows for the definition of a <code>DataSendCompletedEvent</code>
[TPS_SWCT_01337]	<code>DataReceivedEvent</code>
[TPS_SWCT_01338]	<code>DataReceiveErrorEvent</code>
[TPS_SWCT_01339]	RTE activates <code>RunnableEntity</code> in response to <code>DataReceiveErrorEvent</code>
[TPS_SWCT_01340]	<code>DataReceiveErrorEvent</code> cannot be combined with a <code>WaitPoint</code>







Number	Heading
[TPS_SWCT_01341]	<code>DataReceiveErrorEvent</code> is directly associated with the corresponding <code>VariableDataPrototype</code>
[TPS_SWCT_01342]	Invocation of a server operation
[TPS_SWCT_01343]	Synchronous vs. asynchronous invocation
[TPS_SWCT_01344]	Consistency of values of <code>timeout</code>
[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 <code>SwComponentTypes</code>
[TPS_SWCT_01351]	Access to a <code>ParameterDataPrototype</code>
[TPS_SWCT_01352]	Requested mode is just sent and received as an ordinary data value
[TPS_SWCT_01353]	<code>RunnableEntity</code> s react on a mode request via a corresponding <code>RTEEvent</code>
[TPS_SWCT_01354]	<code>PortAPIOption</code>
[TPS_SWCT_01355]	<code>enableTakeAddress = true</code>
[TPS_SWCT_01356]	<code>indirectAPI</code> 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 <code>supportsMultipleInstantiation == false</code>
[TPS_SWCT_01362]	Initialization of <code>PerInstanceMemory</code>
[TPS_SWCT_01363]	<code>PerInstanceMemory</code> typed by 'C' Data Types
[TPS_SWCT_01364]	Initial value of a <code>PerInstanceMemory</code> typed by 'C' Data Types
[TPS_SWCT_01365]	<code>PerInstanceMemory</code> typed by AUTOSAR Data Types
[TPS_SWCT_01366]	Initial value of a <code>PerInstanceMemory</code> 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]	<code>VariationPointProxy</code>
[TPS_SWCT_01371]	<code>VariationPointProxy</code> vs. <code>VariationPoint</code>
[TPS_SWCT_01372]	<code>bindingTime = preCompileTime</code>
[TPS_SWCT_01373]	RTE generator shall evaluate the <code>SwSystemconstDependentFormula</code>
[TPS_SWCT_01374]	Implementation of <code>AutosarParameterRef</code>
[TPS_SWCT_01375]	Implementation of <code>AutosarVariableRef</code>
[TPS_SWCT_01376]	Software-components need to be capable of reacting to state changes





Number	Heading
[TPS_SWCT_01377]	Two mechanisms to define how <code>SwcInternalBehavior</code> should interact with the mode management
[TPS_SWCT_01378]	<code>AtomicSwComponentType</code> can define an <code>SwcModeSwitchEvent</code> to execute <code>RunnableEntity</code>
[TPS_SWCT_01379]	<code>AtomicSwComponentType</code> can indicate whether an <code>RTEEvent</code> that starts an associated <code>RunnableEntity</code> 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]	<code>ModeSwitchPoint</code>
[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 <code>AtomicSwComponentTypes</code> are defined by the <code>initialMode</code>
[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 <code>EcuAbstractionSwComponentType</code> and the corresponding <code>BswModuleDescription</code>
[TPS_SWCT_01393]	Complex Driver
[TPS_SWCT_01394]	Complex Driver is represented by the <code>ComplexDeviceDriverSwComponentType</code>
[TPS_SWCT_01395]	<code>ComplexDeviceDriverSwComponentType</code> has dependencies to ECU Hardware
[TPS_SWCT_01396]	Mapping between the <code>ComplexDeviceDriverSwComponentType</code> and the corresponding <code>BswModuleDescription</code>
[TPS_SWCT_01397]	Hybrid concept between Basic Software Modules and a <code>SwComponentType</code>
[TPS_SWCT_01398]	Communication patterns for AUTOSAR services
[TPS_SWCT_01399]	Dependency is modeled by aggregating required and provided <code>PortPrototypes</code>
[TPS_SWCT_01400]	<code>PortInterface</code> selected from the set of standardized Service Interfaces
[TPS_SWCT_01401]	Form a top-level <code>RootSwCompositionPrototype</code>
[TPS_SWCT_01402]	Mapping of all <code>AtomicSwComponentType</code> instances to <code>EcuInstances</code>
[TPS_SWCT_01403]	Impact of AUTOSAR services on the methodology
[TPS_SWCT_01404]	Creation of the <code>EcuExtract</code>
[TPS_SWCT_01405]	Creation of the <code>ServiceSwComponentTypes</code>
[TPS_SWCT_01406]	Creation of <code>SwComponentPrototype</code> typed by a <code>ServiceSwComponentType</code>





Number	Heading
[TPS_SWCT_01407]	Creation of <code>InternalBehavior</code> typed by a <code>ServiceSwComponentType</code>
[TPS_SWCT_01408]	Creation of <code>SwCbswMapping</code>
[TPS_SWCT_01409]	Update of <code>PortDefinedArgumentValues</code>
[TPS_SWCT_01410]	<code>Dcm</code> and <code>Dem</code> can directly access <code>dataElements</code> in <code>PPortPrototypes</code> typed by a <code>SenderReceiverInterface</code>
[TPS_SWCT_01411]	Use cases for a <code>ServiceSwComponentType</code> to express <code>ServiceNeeds</code>
[TPS_SWCT_01412]	<code>ServiceSwComponentType</code> 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 <code>ServiceProxySwComponentType</code>
[TPS_SWCT_01416]	Difference between a <code>ServiceProxySwComponentType</code> and an <code>ApplicationSwComponentType</code>
[TPS_SWCT_01417]	Define calibration parameters common to all <code>SwComponentPrototypes</code> of the same <code>SwComponentType</code>
[TPS_SWCT_01418]	Ways to define a calibration parameter
[TPS_SWCT_01419]	<code>ParameterSwComponentType</code> shall never aggregate a <code>SwCInternalBehavior</code>
[TPS_SWCT_01420]	<code>SwComponentType</code> requiring access to shared calibration parameters needs <code>RPortPrototype</code> typed by a <code>ParameterInterface</code>
[TPS_SWCT_01421]	<code>ParameterInterface</code> is not restricted to parameters which can actually be calibrated
[TPS_SWCT_01422]	Delegation of <code>PortPrototypes</code> typed by a <code>ParameterInterface</code>
[TPS_SWCT_01423]	<code>ParameterDataPrototype</code> aggregated in the role <code>constantMemory</code>
[TPS_SWCT_01424]	<code>ParameterDataPrototype</code> aggregated in the role <code>perInstanceParameter</code>
[TPS_SWCT_01425]	<code>AtomicSwComponentType</code> provides one callback per event if diagnostic event data change
[TPS_SWCT_01426]	<code>AtomicSwComponentType</code> provides callback if any diagnostic event data and/or status changed
[TPS_SWCT_01427]	<code>AtomicSwComponentType</code> provides data for diagnostic purposes via <code>ClientServerInterface</code>
[TPS_SWCT_01428]	<code>ServiceSwComponentType</code> representing the <code>Dem</code> provides a <code>PPortPrototype</code> for the <code>Dcm</code>
[TPS_SWCT_01429]	[constr_1135] only applies for <code>BITFIELD_TEXTTABLE</code>
[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 <code>CompuScale</code> in C code
[TPS_SWCT_01432]	Keep the <code>invalidValue</code> 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





Number	Heading
[TPS_SWCT_01436]	Different receivers require different handling of data invalidation
[TPS_SWCT_01437]	<code>invalidValue</code> can also be specified without setting a <code>compuMethod</code>
[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 <code>TYPE_REFERENCE</code>
[TPS_SWCT_01442]	<code>ImplementationDataType</code> of <code>category</code> <code>TYPE_REFERENCE</code> does not define own properties
[TPS_SWCT_01443]	<code>ImplementationDataType</code> of <code>category</code> <code>TYPE_REFERENCE</code> overwrites properties of refined <code>ImplementationDataType</code>
[TPS_SWCT_01444]	Size of <code>SwBaseType</code> is specified in bits
[TPS_SWCT_01445]	Applicability of <code>SwDataDefProps</code> for <code>DataPrototypes</code>
[TPS_SWCT_01446]	References to a <code>DataPrototype</code> may or may not imply the necessity for using an <code>instanceRef</code>
[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 <code>category</code> of <code>VariationPointProxy</code>
[TPS_SWCT_02000]	Default value for attribute <code>swImplPolicy</code>
[TPS_SWCT_02001]	Values of <code>SwAxisCont</code> with the <code>category</code> <code>COM_AXIS</code> , <code>RES_AXIS</code> are for display only
[TPS_SWCT_02002]	<code>AtomicSwComponentType</code> offers a <code>PPortPrototypes</code> typed by <code>ClientServerInterface</code> to read/write current value via diagnostic services
[TPS_SWCT_02003]	<code>AtomicSwComponentType</code> offers <code>PortPrototypes</code> typed by <code>SenderReceiverInterfaces</code> to read/write current values via diagnostic services
[TPS_SWCT_02004]	<code>AtomicSwComponentType</code> offers a <code>PortPrototype</code> typed by a <code>ClientServerInterface</code> to start/stop or request routine results of diagnostic routines
[TPS_SWCT_02005]	<code>AtomicSwComponentType</code> offers <code>PortPrototypes</code> typed by <code>ClientServerInterfaces</code> to adjust the IO signal via diagnostic services
[TPS_SWCT_02006]	<code>AtomicSwComponentType</code> offers sender receiver ports to adjust the IO signal via diagnostic services
[TPS_SWCT_02007]	<code>AtomicSwComponentType</code> implements a OBD system monitor with In-Use-Monitor Performance Ratio
[TPS_SWCT_02008]	<code>AtomicSwComponentType</code> offers a server port to read/write current value via OBD services
[TPS_SWCT_02009]	<code>AtomicSwComponentType</code> offers sender receiver ports to read/write current values via OBD services
[TPS_SWCT_02010]	<code>AtomicSwComponentType</code> offers a server port to read vehicle information values via OBD services
[TPS_SWCT_02011]	<code>AtomicSwComponentType</code> offers a server port to read DTR value via OBD services
[TPS_SWCT_02012]	<code>AtomicSwComponentType</code> offers a server port for request control of on-board system, test or component via OBD services





Number	Heading
[TPS_SWCT_02013]	<i>AtomicSwComponentType</i> offers a server port to get protocol, session and security information or to request a Reset to Default Session
[TPS_SWCT_02014]	<i>AtomicSwComponentType</i> supports Response On Event (ROE) via diagnostic services
[TPS_SWCT_02015]	<i>AtomicSwComponentType</i> verifies the access to security level via diagnostic services
[TPS_SWCT_02016]	<i>AtomicSwComponentType</i> requires information on the status of the protocol communication and may disallow a protocol
[TPS_SWCT_02017]	<i>AtomicSwComponentType</i> requires the notification about a Service Request via diagnostic services
[TPS_SWCT_02018]	Setup for <i>AtomicSwComponentType</i> which contains a Supervised Entity
[TPS_SWCT_02019]	Setup for <i>AtomicSwComponentType</i> which requires <i>Global Supervision Status</i> notification
[TPS_SWCT_02020]	<i>AtomicSwComponentType</i> uses the hash calculation of the Crypto Service
[TPS_SWCT_02021]	<i>AtomicSwComponentType</i> uses the message authentication code (MAC) calculation of the Crypto Service
[TPS_SWCT_02022]	<i>AtomicSwComponentType</i> uses the message authentication code (MAC) verification of the Crypto Service
[TPS_SWCT_02023]	<i>AtomicSwComponentType</i> uses the generation of random seed of the Crypto Service
[TPS_SWCT_02024]	<i>AtomicSwComponentType</i> uses the generation of random numbers of the Crypto Service
[TPS_SWCT_02025]	<i>AtomicSwComponentType</i> uses the symmetrical block encryption of the Crypto Service
[TPS_SWCT_02026]	<i>AtomicSwComponentType</i> uses the symmetrical block decryption of the Crypto Service
[TPS_SWCT_02027]	<i>AtomicSwComponentType</i> uses the symmetrical encryption of the Crypto Service
[TPS_SWCT_02028]	<i>AtomicSwComponentType</i> uses the symmetrical decryption of the Crypto Service
[TPS_SWCT_02029]	<i>AtomicSwComponentType</i> uses the asymmetrical encryption of the Crypto Service
[TPS_SWCT_02030]	<i>AtomicSwComponentType</i> uses the asymmetrical decryption of the Crypto Service
[TPS_SWCT_02031]	<i>AtomicSwComponentType</i> uses the signature generation of the Crypto Service
[TPS_SWCT_02032]	<i>AtomicSwComponentType</i> uses the signature verification of the Crypto Service
[TPS_SWCT_02033]	<i>AtomicSwComponentType</i> uses the checksum calculation of the Crypto Service
[TPS_SWCT_02034]	<i>AtomicSwComponentType</i> uses the key derivation of the Crypto Service
[TPS_SWCT_02035]	<i>AtomicSwComponentType</i> uses the symmetric key derivation of the Crypto Service
[TPS_SWCT_02036]	<i>AtomicSwComponentType</i> uses the key exchange interface for public value calculation of the Crypto Service





Number	Heading
[TPS_SWCT_02037]	<a href="#">AtomicSwComponentType</a> uses the key exchange interface for secret value calculation of the Crypto Service
[TPS_SWCT_02038]	<a href="#">AtomicSwComponentType</a> uses the key exchange interface to calculate symmetric key with the Crypto Service
[TPS_SWCT_02039]	<a href="#">AtomicSwComponentType</a> uses the symmetrical key extraction of the Crypto Service
[TPS_SWCT_02040]	<a href="#">AtomicSwComponentType</a> uses the symmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a symmetric key
[TPS_SWCT_02041]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a asymmetric key
[TPS_SWCT_02042]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical public key extraction of the Crypto Service
[TPS_SWCT_02043]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical private key extraction of the Crypto Service
[TPS_SWCT_02044]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a symmetrical wrapping key
[TPS_SWCT_02045]	<a href="#">AtomicSwComponentType</a> 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 <a href="#">Units</a> in <a href="#">CompuMethods</a> (the text is still there but it does no longer represent a constraint)
[constr_1062]	Compatibility of data types with <a href="#">category</a> BIT
[constr_1122]	Existence of attributes in PROFILE_03
[constr_1123]	Constraints of <a href="#">dataLength</a> in PROFILE_03
[constr_1124]	Constraints of <a href="#">dataId</a> in PROFILE_03
[constr_1125]	Constraints of <a href="#">maxDeltaCounterInit</a> in PROFILE_03
[constr_1127]	<a href="#">ServiceSwComponentType</a> shall not have <a href="#">ServiceNeeds</a>
[constr_1136]	Compatibility of <a href="#">introduction</a> of blueprint and blueprinted element
	The following constraints are moved to [1]
[constr_2500]	<a href="#">PortInterfaces</a> shall be of same kind
[constr_2526]	<a href="#">PortInterfaces</a> need to be compatible to the blueprints
[constr_2527]	Blueprints shall live in package of a proper category
[constr_2528]	<a href="#">PortPrototypes</a> shall not refer to blueprints of <a href="#">PortInterfaces</a>
[constr_2529]	Blueprints of ports and interfaces shall be compatible





Number	Heading
[constr_4001]	Content of <a href="#">ModeRequestTypeMap</a>

**Table C.8: Deleted Constraints in R4.0.3**

### C.3.5 Deleted Specification Items in R4.0.3

N/A

## C.4 Constraint History of this Document according to AUTOSAR R4.1.1

### C.4.1 Changed Constraints in R4.1.1

Number	Heading
[constr_1012]	Value of <a href="#">category</a> is <code>FIXED_LENGTH</code>
[constr_1013]	Value of <a href="#">category</a> is <code>VARIABLE_LENGTH</code>
[constr_1016]	Restriction of <a href="#">invalidValue</a> for <a href="#">ImplementationDataType</a> and <a href="#">ImplementationDataTypeElement</a>
[constr_1026]	Compatibility of <a href="#">Units</a>
[constr_1047]	Compatibility of <a href="#">ApplicationPrimitiveDataTypes</a>
[constr_1048]	Compatibility of <a href="#">ApplicationRecordDataTypes</a>
[constr_1049]	Compatibility of <a href="#">ApplicationArrayDataTypes</a>
[constr_1050]	Compatibility of <a href="#">ImplementationDataTypes</a>
[constr_1060]	Compatibility of data types with <a href="#">category</a> <code>ARRAY</code> , <code>VAL_BLK</code>
[constr_1072]	Compatibility of <a href="#">ModeSwitchInterfaces</a> in the context of an <a href="#">AssemblySwConnector</a>
[constr_1073]	Compatibility of <a href="#">ModeSwitchInterfaces</a> in the context of an <a href="#">DelegationSwConnector</a>
[constr_1074]	Compatibility of <a href="#">ModeDeclarationGroupPrototypes</a>
[constr_1075]	Compatibility of <a href="#">ModeDeclarationGroups</a>
[constr_1079]	Compatibility of <a href="#">ClientServerInterfaces</a> in the context of an <a href="#">AssemblySwConnector</a>
[constr_1080]	Compatibility of <a href="#">ClientServerInterfaces</a> in the context of an <a href="#">DelegationSwConnector</a>
[constr_1081]	Compatibility of <a href="#">TriggerInterfaces</a> in the context of an <a href="#">AssemblySwConnector</a>
[constr_1082]	Compatibility of <a href="#">TriggerInterfaces</a> in the context of an <a href="#">DelegationSwConnector</a>
[constr_1068]	Compatibility of <a href="#">VariableDataPrototypes</a> or <a href="#">ParameterDataPrototypes</a> typed by primitive data types





Number	Heading
[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 <code>VariableDataPrototypes</code> or <code>ParameterDataPrototypes</code> 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 <code>Limit</code> shall yield a numerical value
[constr_1192]	Compatibility of “IDENTICAL” to “RAT_FUNC” or “LINEAR”
[constr_1193]	<code>ModeDeclaration</code> shall be referenced by at least one <code>ModeTransition</code> in the role <code>enteredMode</code>
[constr_1194]	Identical <code>ModeTransitions</code>
[constr_1195]	<code>SwcModeSwitchEvent</code> and the definition of <code>ModeTransition</code>
[constr_1196]	Existence of <code>networkRepresentation</code> vs. <code>compositeNetworkRepresentation</code>
[constr_1197]	Existence of <code>compositeNetworkRepresentation</code> shall be comprehensive
[constr_1200]	Queued communication is not applicable for <code>dataElements</code> owned by <code>PRPortPrototype</code>
[constr_1201]	<code>initValue</code> shall exist in an <code>RPortPrototype</code>
[constr_1202]	Supported connections by <code>AssemblySwConnector</code> for <code>PortPrototypes</code> typed by a <code>SenderReceiverInterface</code> or <code>NvDataInterface</code>
[constr_1203]	Supported connections by <code>DelegationSwConnector</code> for <code>PortPrototypes</code> typed by a <code>SenderReceiverInterface</code> or <code>NvDataInterface</code>







Number	Heading
[constr_1204]	Supported connections by <code>AssemblySwConnector</code> for <code>PortPrototypes</code> typed by a <code>ClientServerInterface</code> , <code>ModeSwitchInterface</code> , or <code>TriggerInterface</code>
[constr_1205]	Supported connections by <code>DelegationSwConnector</code> for <code>PortPrototypes</code> typed by a <code>ClientServerInterface</code> , <code>ModeSwitchInterface</code> , or <code>TriggerInterface</code>
[constr_1209]	Mapping of <code>ModeDeclarations</code> of mode user to <code>ModeDeclaration</code> of mode manager
[constr_1210]	Mapping of <code>ModeDeclarations</code> of mode user to <b>all</b> <code>ModeDeclarations</code> of mode manager
[constr_1211]	Constraints of <code>maxNoNewOrRepeatedData</code> in PROFILE_01
[constr_1212]	Constraints of <code>syncCounterInit</code> in PROFILE_01
[constr_1213]	Constraints of <code>maxNoNewOrRepeatedData</code> in PROFILE_02
[constr_1214]	Constraints of <code>syncCounterInit</code> in PROFILE_02
[constr_1215]	Interpretation of attribute <code>maxNoNewOrRepeatedData</code> owned by <code>EndToEndDescription</code> in PROFILE_01
[constr_1216]	Interpretation of attribute <code>syncCounterInit</code> owned by <code>EndToEndDescription</code> in PROFILE_01
[constr_1217]	Interpretation of attribute <code>maxNoNewOrRepeatedData</code> owned by <code>EndToEndDescription</code> in PROFILE_02
[constr_1218]	Interpretation of attribute <code>syncCounterInit</code> owned by <code>EndToEndDescription</code> in PROFILE_02
[constr_1219]	Invalidation depends on the value of <code>swImplPolicy</code>
[constr_1220]	Compatibility of <code>SwBaseType</code>
[constr_1221]	<code>DataPrototype</code> is typed by an <code>ApplicationPrimitiveDataType</code>
[constr_1222]	<code>category</code> of an <code>AutosarDataType</code> used to type a <code>DataPrototype</code> is set to STRING
[constr_1223]	<code>DataPrototype</code> is typed by an <code>ApplicationRecordDataType</code>
[constr_1224]	<code>DataPrototype</code> is typed by an <code>ApplicationArrayDataType</code>
[constr_1225]	<code>DataPrototype</code> is typed by an <code>ImplementationDataType</code> that references a <code>CompuMethod</code> of category TEXTTABLE or BITFIELD_TEXTTABLE
[constr_1226]	Applicable range for <code>ExecutableEntityActivationReason.bitPosition</code>
[constr_1227]	Value of attribute <code>ExecutableEntityActivationReason.bitPosition</code> shall be unique
[constr_1228]	<code>RTEEvent</code> that is referenced by a <code>WaitPoint</code> in the role <code>trigger</code> shall not reference <code>ExecutableEntityActivationReason</code>
[constr_1229]	<code>category</code> of <code>ImplementationDataType</code> boils down to VALUE
[constr_1230]	<code>ApplicationDataType</code> that qualifies for Integral Primitive Type
[constr_1231]	<code>ConsistencyNeeds</code> aggregated by <code>CompositionSwComponentType</code>
[constr_1232]	<code>ConsistencyNeeds</code> aggregated by <code>AtomicSwComponentType</code>
[constr_1233]	<code>InstantiationTimingEventProps</code> shall only reference <code>TimingEvent</code>
[constr_1234]	Value of <code>RunnableEntity.symbol</code>
[constr_1237]	Scope of mapped <code>ClientServerOperations</code> in the context of a <code>ClientServerOperationMapping</code>





Number	Heading
[constr_1238]	Scope of mapped <code>ApplicationErrors</code> in the context of a <code>ClientServerOperationMapping</code>
[constr_1239]	<code>RuleBasedValueSpecification</code> shall not exceed the number of values required
[constr_1240]	Consistency of <code>ArgumentDataPrototypes</code> within the context of a <code>ClientServerOperationMapping</code>
[constr_1241]	Compound Primitive Data Types and <code>invalidValue</code>
[constr_1242]	Restriction of <code>invalidValue</code> for <code>ApplicationPrimitiveDataType</code>
[constr_1243]	<code>NumericalOrText</code> shall either define <code>vf</code> or <code>vt</code>
[constr_1244]	<code>DataPrototypes</code> used in application software shall not be typed by C enums
[constr_1245]	Consideration of <code>ModeTransitions</code> for the compatibility of <code>ModeDeclarationGroups</code>
[constr_1246]	Consistency of <code>firstMode</code> and <code>secondMode</code> in the scope of one <code>ModeDeclarationMappingSet</code>
[constr_1247]	Consistency of <code>ModeDeclarationMappingSet</code> with respect to the referenced <code>firstModeGroup</code> and <code>secondModeGroup</code>
[constr_1248]	Compatibility of <code>PortPrototypes</code> of different <code>DataInterfaces</code> in the context of a <code>PassThroughSwConnector</code>
[constr_1249]	Compatibility of <code>ModeSwitchInterfaces</code> in the context of a <code>PassThroughSwConnector</code>
[constr_1250]	Compatibility of <code>ClientServerInterfaces</code> in the context of a <code>PassThroughSwConnector</code>
[constr_1251]	Compatibility of <code>PortPrototypes</code> of <code>TriggerInterfaces</code> in the context of a <code>PassThroughSwConnector</code>
[constr_1252]	Creation of a loop involving a <code>PassThroughSwConnector</code> is not allowed
[constr_1253]	Supported usage of <code>VariationPointProxy</code>
[constr_1254]	Definition of a pointer to a pointer
[constr_1255]	<code>ApplicationPrimitiveDataTypes</code> of category <code>BOOLEAN</code> and <code>STRING</code>
[constr_1256]	Acknowledgement feedback in n:1 writer case
[constr_1257]	No <code>WaitPoints</code> allowed
[constr_1258]	Value of <code>minimumStartInterval</code> for <code>RunnableEntitys</code> triggered by an <code>InitEvent</code>
[constr_1259]	Aggregation of <code>AsynchronousServerCallPoint</code> and <code>AsynchronousServerCallResultPoint</code>
[constr_1260]	No mode disabling for <code>InitEvents</code>
[constr_1261]	Applicability for <code>EndToEndDescription.dataIdNibbleOffset</code>
[constr_1263]	Existence of <code>ModeErrorBehavior.defaultMode</code>
[constr_1264]	Iteration along output axis is only supported for <code>VALUE</code> and <code>VAL_BLK</code>
[constr_1268]	<code>ArgumentDataPrototype.direction</code> shall be preserved in a <code>ClientServerOperationMapping</code>
[constr_1269]	Number of <code>arguments</code> shall be preserved in a <code>ClientServerOperationMapping</code>
[constr_1270]	<code>ArgumentDataPrototype</code> shall be mapped only once in a <code>ClientServerOperationMapping</code>





Number	Heading
[constr_1271]	<code>ArrayValueSpecification.elements</code> shall be identical to the number of <code>ApplicationRecordDataType.element</code>
[constr_1272]	<code>ArrayValueSpecification.elements</code> shall be identical to the number of <code>subElements</code> of <code>ImplementationDataType</code> of category STRUCTURE
[constr_1273]	<code>ArrayValueSpecification.elements</code> shall be identical to the value of <code>ApplicationArrayDataType.element.maxNumberOfElements</code>
[constr_1274]	<code>ArrayValueSpecification.elements</code> shall be identical to the value of <code>ImplementationDataType.subElement.arraySize</code> of category ARRAY
[constr_2054]	Valid targets of <code>rptSystem</code>
[constr_2055]	Valid targets of <code>byPassPoint</code> and <code>rptHook</code> reference
[constr_2056]	Consistency of <code>RapidPrototypingScenario</code> with respect to <code>rptSystem</code> and <code>rptArHook</code> references
[constr_2057]	Mandatory information of a <code>RuleBasedAxisCont</code>
[constr_2058]	Mandatory information of a <code>RuleBasedValueCont</code>
[constr_4082]	<code>RunnableEntity.reentrancyLevel</code> 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 <code>symbol</code> of the <code>symbolProps</code>
[TPS_SWCT_01001]	Prefix symbols generated for the <code>RunnableEntity</code>
[TPS_SWCT_01085]	Variation on the behavior level
[TPS_SWCT_01112]	Semantics of <code>PortPrototypes</code>
[TPS_SWCT_01113]	Connecting two <code>PortPrototypes</code>
[TPS_SWCT_01128]	<code>SwRecordLayout</code> needed for <code>ApplicationPrimitiveDataType</code> 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 <code>category</code> of <code>VariationPointProxy</code>
[TPS_SWCT_01449]	Semantics of a <code>ModeDeclarationGroupPrototypeMapping</code>
[TPS_SWCT_01450]	Semantics of a <code>ModeTransition</code>
[TPS_SWCT_01451]	Relations between <code>ModeTransition</code> and <code>ModeDeclaration</code>
[TPS_SWCT_01452]	Applicability of <code>networkRepresentation</code> for <code>ApplicationCompositeDataType</code>
[TPS_SWCT_01454]	<code>PRPortPrototype</code> can own both <code>RPortComSpecs</code> and <code>PPortComSpecs</code>
[TPS_SWCT_01455]	Duplicate existence of <code>initValue</code> in the context of a <code>PRPortPrototype</code>
[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>ModeDeclarationGroupPrototypeMapping.modeDeclarationMappingSet</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>regRequiresStability</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>regRequiresStability</code>
[TPS_SWCT_01475]	Sending <code>SwComponentType</code> owns a <code>DataPrototypeGroup</code> in the role <code>regRequiresStability</code>
[TPS_SWCT_01476]	Sender and receiver of the same implicitly communicated <code>VariableDataPrototypes</code> are associated with the same <code>RunnableEntityGroup</code>





Number	Heading
[TPS_SWCT_01477]	Integral Primitive Types
[TPS_SWCT_01478]	Array size is defined as an attribute of the <a href="#">ImplementationDataTypeElement</a>
[TPS_SWCT_01479]	Applicability of <a href="#">ConsistencyNeeds</a>
[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 <a href="#">ConsistencyNeeds</a>
[TPS_SWCT_01482]	The meaning of the term <i>coherence</i> with respect to <a href="#">ConsistencyNeeds</a>
[TPS_SWCT_01483]	Use static and constant memory to support Measurement and Calibration
[TPS_SWCT_01484]	Meaning of <a href="#">ApplicationRuleBasedValueSpecification</a>
[TPS_SWCT_01485]	<a href="#">RuleBasedValueSpecification</a> shall initialize first elements in an array
[TPS_SWCT_01486]	<a href="#">ApplicationPrimitiveDataType</a> of category STRING may have <a href="#">invalidValue</a>
[TPS_SWCT_01487]	Correspondence of <a href="#">invalidValue</a> for <a href="#">ApplicationPrimitiveDataType</a> and <a href="#">ImplementationDataType</a>
[TPS_SWCT_01488]	<a href="#">ApplicationPrimitiveDataType</a> shall be interpreted as a string of a particular encoding
[TPS_SWCT_01489]	Standardized values of <a href="#">SwRecordLayoutV.swRecordLayoutVProp</a>
[TPS_SWCT_01490]	AUTOSAR supports <a href="#">ApplicationErrors</a> only for <a href="#">ClientServerInterfaces</a>
[TPS_SWCT_01491]	AUTOSAR system does not need to explicitly describe infrastructure errors
[TPS_SWCT_01492]	Default values for <a href="#">factorSiToUnit</a> and <a href="#">offsetSiToUnit</a>
[TPS_SWCT_01493]	The number of <a href="#">RuleArguments.arguments</a> shall not exceed the array size
[TPS_SWCT_01494]	A <a href="#">RuleBasedValueSpecification</a> of rule <a href="#">FILL_UNITIL_END</a> shall fill the value of the last <a href="#">RuleArguments.argument</a> until the last element of the array
[TPS_SWCT_01495]	Standardized value of <a href="#">RuleBasedValueSpecification.category</a>
[TPS_SWCT_01496]	General precedence rule for attributes of <a href="#">SwDataDefProps</a>
[TPS_SWCT_01497]	Precedence of the <a href="#">unit</a> of value axis
[TPS_SWCT_01498]	Precedence of the <a href="#">DataConstr</a> of value axis
[TPS_SWCT_01499]	Precedence of the <a href="#">CompuMethod</a> 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 <a href="#">Unit</a> of the input axis
[TPS_SWCT_01503]	Precedence of the <a href="#">DataConstr</a> 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 <a href="#">PassThroughSwConnector</a>
[TPS_SWCT_01508]	Scope of end-to-end protection
[TPS_SWCT_01509]	Implicit communication behavior
[TPS_SWCT_01510]	The role of pretended networking





Number	Heading
[TPS_SWCT_01511]	Configuration option is encoded into <a href="#">ModeDeclaration</a>
[TPS_SWCT_01512]	Request change of <code>Pretended Networking mode</code>
[TPS_SWCT_01513]	React on the change of <code>Pretended Networking mode</code>
[TPS_SWCT_01514]	Duplicate existence of <code>initValue</code> in the context of a <a href="#">PRPortPrototype</a>
[TPS_SWCT_01515]	<a href="#">PPortInCompositionInstanceRef</a> shall be used for attaching <a href="#">DelegationSwConnector</a> to an inner <a href="#">PRPortPrototype</a>
[TPS_SWCT_01516]	<a href="#">PortInterface</a> describes the static structure of information interchange
[TPS_SWCT_01517]	<a href="#">ClientServerOperation</a> 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 <a href="#">RunnableEntity</a> periodically
[TPS_SWCT_01520]	Implication of the existence of <a href="#">possibleError</a> on compatibility of <a href="#">ClientServerOperations</a>
[TPS_SWCT_01521]	Use <a href="#">AutosarVariableRef.localVariable</a> for referencing inter-runnable variables
[TPS_SWCT_01522]	No initial value is specified for <a href="#">implicitInterRunnableVariable</a> or <a href="#">explicitInterRunnableVariable</a>
[TPS_SWCT_01523]	Internal trigger event
[TPS_SWCT_01524]	Usage of <a href="#">IoHwAbstractionServerAnnotation</a>
[TPS_SWCT_01525]	<a href="#">InitEvent</a> references a <a href="#">RunnableEntity</a> in the role <code>startOnEvent</code>
[TPS_SWCT_01528]	Meaning of <a href="#">NumericalRuleBasedValueSpecification</a>
[TPS_SWCT_01529]	Default value for <a href="#">EndToEndDescription.dataIdNibbleOffset</a>
[TPS_SWCT_01530]	Error behavior of mode manager and mode user
[TPS_SWCT_01531]	The semantics of <a href="#">ModeErrorReactionPolicyEnum</a>
[TPS_SWCT_01532]	The role of <a href="#">ModeErrorBehavior.defaultMode</a>
[TPS_SWCT_01533]	<a href="#">ModeDeclarationGroup.initialMode</a> shall be assumed in the absence of <a href="#">ModeDeclarationGroup.modeManagerErrorBehavior</a>
[TPS_SWCT_01534]	<a href="#">ModeDeclarationGroup.initialMode</a> shall be assumed in the absence of <a href="#">ModeDeclarationGroup.modeUserErrorBehavior</a>
[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 <a href="#">modeUserErrorBehavior</a>
[TPS_SWCT_01542]	Preferential selection of <a href="#">modeManagerErrorBehavior</a>
[TPS_SWCT_01543]	<a href="#">PortInterfaceMapping</a> overrides all other compatibility rules
[TPS_SWCT_01544]	<i>prefix</i> used for the actual name of the used <a href="#">PortInterface</a> for the routing activation
[TPS_SWCT_01545]	<a href="#">ModeDeclaration</a> of a <i>mode user</i> that is not mapped to a <a href="#">ModeDeclaration</a> 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]	<a href="#">byPassPoint</a> specifies the rapid prototyping capability





Number	Heading
[TPS_SWCT_02047]	<a href="#">RptHook</a> specifies the link to rapid prototyping algorithm
[TPS_SWCT_02048]	Implicit <a href="#">SwComponentPrototype</a> selection for Rapid Prototyping Scenario
[TPS_SWCT_02049]	Implicit <a href="#">RunnableEntity</a> selection for Rapid Prototyping Scenario
[TPS_SWCT_02050]	Explicit access point selection for Rapid Prototyping Scenario
[TPS_SWCT_02051]	Explicit <a href="#">DataPrototype</a> selection for Rapid Prototyping Scenario
[TPS_SWCT_02052]	Definition of Rapid Prototyping Scenario is splittable
[TPS_SWCT_02053]	Values of <a href="#">RuleBasedAxisCont</a> with the <a href="#">category</a> COM_AXIS, RES_AXIS are for display only
[TPS_SWCT_02507]	Instantiation-specific <a href="#">RTEEvents</a>

**Table C.12: Added Specification Items in 4.1.1**

#### C.4.5 Deleted Constraints in R4.1.1

Number	Heading
[constr_1239]	<a href="#">RuleBasedValueSpecification</a> shall not exceed the number of values required
[constr_1145]	Finding the symbol for the representation of a <a href="#">CompuScale</a> 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]	<a href="#">IoHwAbstractionServerAnnotation</a>

**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 <code>category</code> <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 <code>categorys</code> for attribute <code>ImplementationDataType.swDataDef-Props.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>







Number	Heading
[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. <code>category</code> for <code>DataPrototypes</code> typed by <code>ImplementationDataTypes</code>
[constr_1289]	Allowed Attributes vs. <code>category</code> 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 <code>category</code> DATA_REFERENCE
[constr_1296]	<code>DataPrototypes</code> used as <code>explicitInterRunnableVariable</code> or <code>implicitInterRunnableVariable</code> and <code>category</code> DATA_REFERENCE

**Table C.16: Added Constraints in R4.1.2**

### C.5.3 Changed Specification Items in R4.1.2

Number	Heading
[TPS_SWCT_01006]	<code>ImplementationDataType.subElement.arraySize</code> shall be used to define the size of the array
[TPS_SWCT_01175]	Actively query the status of a partial network
[TPS_SWCT_01219]	<code>ComSpec</code> for queued and non-queued sender-receiver communication
[TPS_SWCT_01154]	Attribute <code>prefix</code> of <code>IncludedModeDeclarationGroupSet</code>
[TPS_SWCT_02011]	<code>AtomicSwComponentType</code> 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





Number	Heading
[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]	<code>ModeSwitchedAckEvent</code> is triggered by the RTE regardless
[TPS_SWCT_01556]	Rule for setting <code>RoleBasedPortAssignment.role</code>
[TPS_SWCT_01557]	<code>dataWriteAccess</code> also allows for the definition of a <code>DataWriteCompletedEvent</code>
[TPS_SWCT_01558]	<code>DataWriteCompletedEvent</code> cannot be combined with a <code>WaitPoint</code>
[TPS_SWCT_01559]	Default value for attribute <code>SwDataDefProps.swCalibrationAccess</code>
[TPS_SWCT_01560]	Supported <code>categorys</code> of <code>CompuMethods</code> for data conversion
[TPS_SWCT_01561]	Application of data conversion to composite <code>AutosarDataTypes</code>
[TPS_SWCT_01562]	Specification of values of an enumeration
[TPS_SWCT_01563]	Applicable values for <code>nativeDeclaration</code>
[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 <code>FlatInstanceDescriptor</code>
[TPS_SWCT_01567]	Default behavior for <code>invalidationPolicy</code>
[TPS_SWCT_01568]	Consideration of <code>RPortComSpec</code> or <code>PPortComSpec</code> depending on the ownership
[TPS_SWCT_01569]	Definition of <code>CompuScale</code> Symbolic Name
[TPS_SWCT_01570]	<code>DataTypeMap</code> is mandatory in the presence of <code>ApplicationPrimitiveDataType.swDataDefProps.swRecordLayout</code>

**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 <code>symbol</code> of <code>RunnableEntity</code>
[constr_2025]	Uniqueness of <code>symbol</code> attributes (This constraint has been relocated to [10])
[constr_2032]	<code>transmissionAcknowledge</code> requires a <code>DataSendCompletedEvent</code>
[constr_4011]	<code>ComSpec</code> and <code>ModeSwitchedAckEvent</code>

**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 R4.1.3

Number	Heading
[TPS_SWCT_01573]	A <code>PRPortPrototype</code> is never considered unconnected
[TPS_SWCT_01574]	<code>PerInstanceMemory.typeDefinition</code> shall not contain a function pointer

**Table C.21: Added Traceables in 4.1.3**

### C.6.2 Changed Traceables in R4.1.3

Number	Heading
[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_01049]	Two ways to use the <code>ExclusiveAreas</code>

**Table C.22: Changed Traceables in R4.1.3**

### C.6.3 Deleted Traceables in R4.1.3

Number	Heading
[TPS_SWCT_01417]	Define calibration parameters common to all <code>SwComponentPrototypes</code> of the same <code>SwComponentType</code>
[TPS_SWCT_01419]	<code>ParameterSwComponentType</code> shall never aggregate a <code>SwcInternalBehavior</code>
[TPS_SWCT_02006]	<code>AtomicSwComponentType</code> offers sender receiver ports to adjust the IO signal via diagnostic services

**Table C.23: Deleted Traceables in R4.1.3**

### C.6.4 Added Constraints in R4.1.3

Number	Heading
[constr_1297]	Applicability of <a href="#">serverArgumentImplPolicy</a> set to [useArrayBaseType]
[constr_1298]	Existence of attributes if <a href="#">category</a> of a <a href="#">ModeDeclarationGroup</a> is set to EXPLICIT_ORDER
[constr_1299]	Existence of attributes if <a href="#">category</a> of a <a href="#">ModeDeclarationGroup</a> is set to other than EXPLICIT_ORDER

**Table C.24: Added Constraints in R4.1.3**

### C.6.5 Changed Constraints in R4.1.3

Number	Heading
[constr_1006]	applicable data categories
[constr_1007]	Allowed attributes of <a href="#">SwDataDefProps</a> for <a href="#">ApplicationDataTypes</a>
[constr_1230]	<a href="#">ApplicationDataType</a> that qualifies for Integral Primitive Type

**Table C.25: Changed Constraints in R4.1.3**

### C.6.6 Deleted Constraints in R4.1.3

Number	Heading
[constr_1179]	Existence of <a href="#">ModeDeclaration.value</a> within a <a href="#">ModeDeclarationGroup</a>
[constr_1180]	Existence of <a href="#">ModeDeclarationGroup.onTransitionValue</a>

**Table C.26: Deleted Constraints in R4.1.3**

## C.7 Constraint History of this Document according to AUTOSAR R4.2.1

### C.7.1 Added Traceables in R4.2.1

Number	Heading
[TPS_SWCT_01572]	Application Mode Manager interacts with both <a href="#">BswM</a> and other <a href="#">Application-SwComponentTypes</a>
[TPS_SWCT_01577]	<a href="#">AtomicSwComponentType</a> requires the notification about a Service Request via diagnostic services with <i>manufacturer</i> characteristics





Number	Heading
[TPS_SWCT_01578]	<code>AtomicSwComponentType</code> requires the notification about a Service Request via diagnostic services with <i>supplier</i> characteristics
[TPS_SWCT_01579]	Dcm can directly access <code>dataElements</code> in <code>PPortPrototypes</code> typed by a <code>SenderReceiverInterface</code>
[TPS_SWCT_01580]	Dem can directly access <code>dataElements</code> in <code>PPortPrototypes</code> typed by a <code>SenderReceiverInterface</code>
[TPS_SWCT_01581]	Communication patterns for mode-related communication
[TPS_SWCT_01582]	Semantics of <code>DiagnosticEventNeeds.deferringFid</code>
[TPS_SWCT_01583]	Completeness of <code>TextTableMapping</code> is not a requirement
[TPS_SWCT_01584]	<code>InternalBehavior</code> of a <code>NvBlockSwComponentType</code> 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>





Number	Heading
[TPS_SWCT_01610]	Modeling of a <a href="#">Variable-Size Array Data Type with Size Indicator</a> enabled
[TPS_SWCT_01611]	Enable <a href="#">Size Indicator</a>
[TPS_SWCT_01612]	<a href="#">arraySizeHandling</a> specifies how the size is determined
[TPS_SWCT_01613]	Internal structure of mapped <a href="#">ImplementationDataType</a>
[TPS_SWCT_01614]	Profiles for internal structure of mapped <a href="#">ImplementationDataType</a>
[TPS_SWCT_01615]	Custom profiles for internal structure of mapped <a href="#">ImplementationDataType</a>
[TPS_SWCT_01616]	Semantics of <a href="#">TransformerHardErrorEvent</a>
[TPS_SWCT_01617]	Structure of an <a href="#">ImplementationDataType</a> that represents a variable-sized array data type
[TPS_SWCT_01618]	<a href="#">Size Indicator</a> for <a href="#">dynamicArraySizeProfile</a> set to <a href="#">VSA_LINEAR</a> , <a href="#">VSA_SQUARE</a> , or <a href="#">VSA_FULLY_FLEXIBLE</a>
[TPS_SWCT_01619]	<a href="#">Size Indicator</a> for <a href="#">dynamicArraySizeProfile</a> set to <a href="#">VSA_RECTANGULAR</a>
[TPS_SWCT_01620]	<a href="#">Size Indicator</a> for <a href="#">dynamicArraySizeProfile</a> set to <a href="#">VSA_RECTANGULAR</a>
[TPS_SWCT_01621]	Payload for <a href="#">dynamicArraySizeProfile</a>
[TPS_SWCT_01622]	Modeling of a <a href="#">Variable-Size Array Data Type only with ImplementationDataType</a>
[TPS_SWCT_01623]	Justification for the existence of attributes <a href="#">ApplicationArrayDataType</a> , <a href="#">dynamicArraySizeProfile</a> and <a href="#">ApplicationArrayElement.arraySizeHandling</a>
[TPS_SWCT_01624]	Hard error occurs during the execution of a transformer chain
[TPS_SWCT_01625]	Sending <a href="#">SwComponentType</a> owns a <a href="#">DataPrototypeGroup</a> in the role <a href="#">dp-gRequiresCoherency</a> and also <a href="#">RunnableEntityGroups</a>
[TPS_SWCT_01626]	Error notification of data transformer errors
[TPS_SWCT_01627]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Security Access
[TPS_SWCT_01628]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Data Services
[TPS_SWCT_01629]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Data Services
[TPS_SWCT_01630]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Data Services
[TPS_SWCT_01631]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Infotype Services
[TPS_SWCT_01632]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Routine Services
[TPS_SWCT_01633]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Request Control Services
[TPS_SWCT_01634]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Data Services
[TPS_SWCT_01635]	Naming conventions may support the effectiveness of <a href="#">SymbolProps</a>





Number	Heading
[TPS_SWCT_01636]	Definition of profiles for the definition of <a href="#">Variable-Size Array Data Types</a>

**Table C.27: Added Traceables in 4.2.1**

### C.7.2 Changed Traceables in R4.2.1

Number	Heading
[TPS_SWCT_01044]	<a href="#">ServiceNeeds</a>
[TPS_SWCT_01053]	Relationship of interchanged data with <a href="#">RunnableEntitys</a>
[TPS_SWCT_01141]	<a href="#">AtomicSwComponentType</a> may have <a href="#">AbstractRequiredPortPrototypes</a> typed by an <a href="#">NvDataInterface</a>
[TPS_SWCT_01150]	<a href="#">InternalBehavior</a> of a <a href="#">NvBlockSwComponentType</a> to enable access to the NVRAM Block management API
[TPS_SWCT_01162]	Existence of <a href="#">TextTableMapping</a>
[TPS_SWCT_01180]	Maximum possible size of Compound Primitive Data Type
[TPS_SWCT_01227]	Unconnected <a href="#">AbstractRequiredPortPrototype</a> typed by <a href="#">NvDataInterface</a>
[TPS_SWCT_01228]	<a href="#">NvProvideComSpec</a>
[TPS_SWCT_01239]	default value for attribute <a href="#">category</a> used in the context of <a href="#">SwSystemconst</a>
[TPS_SWCT_01274]	<a href="#">SwDataDefProps</a> used to support calibration and measurement
[TPS_SWCT_01321]	Communication among <a href="#">RunnableEntitys</a>
[TPS_SWCT_01330]	<a href="#">RunnableEntity</a> can also have <a href="#">dataSendPoints</a>
[TPS_SWCT_01335]	Combine <a href="#">dataReceivePointByValue</a> or <a href="#">dataReceivePointByArgument</a> with a <a href="#">WaitPoint</a>
[TPS_SWCT_01348]	Trigger source
[TPS_SWCT_01385]	Execution of finalization code for software-components
[TPS_SWCT_01456]	Predefined values for <a href="#">MemorySection.option</a> and <a href="#">SwAddrMethod.option</a>
[TPS_SWCT_01475]	Sending <a href="#">SwComponentType</a> owns a <a href="#">DataPrototypeGroup</a> in the role <a href="#">dp-gRequiresCoherency</a>
[TPS_SWCT_01494]	A <a href="#">RuleBasedValueSpecification</a> of rule <code>FILL_UNTIL_END</code> shall fill the value of the last <a href="#">RuleArguments.argument</a> until the last element of the array
[TPS_SWCT_01495]	Standardized value of <a href="#">RuleBasedValueSpecification.rule</a>
[TPS_SWCT_01549]	Definition of linear data scaling
[TPS_SWCT_01560]	Supported <a href="#">categorys</a> of <a href="#">CompuMethods</a> 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 <a href="#">NvBlockSwComponentType</a>





Number	Heading
[TPS_SWCT_02504]	Setup for Nvm Use Case: RAM Block with explicit synchronization using Mirror Interfaces

**Table C.28: Changed Traceables in R4.2.1**

### C.7.3 Deleted Traceables in R4.2.1

Number	Heading
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]	<code>Dcm</code> and <code>Dem</code> 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 R4.2.1**

### C.7.4 Added Constraints in R4.2.1

Number	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>







Number	Heading
[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]	<code>PortPrototypes</code> typed by a <code>ParameterInterface</code>
[constr_1313]	Completeness of <code>TextTableMapping</code> for the values of a given bit mask on the sender side
[constr_1314]	Profile <code>VSA_LINEAR</code> for <code>ApplicationArrayDataType</code>
[constr_1315]	Profile <code>VSA_SQUARE</code> for <code>ApplicationArrayDataType</code>
[constr_1316]	Profile <code>VSA_RECTANGULAR</code> for <code>ApplicationArrayDataType</code>
[constr_1317]	Profile <code>VSA_FULLY_FLEXIBLE</code> for <code>ApplicationArrayDataType</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 <code>VSA_FULLY_FLEXIBLE</code> for <code>ImplementationDataType</code>
[constr_1322]	<code>Size Indicator</code> for undefined <code>dynamicArraySizeProfile</code>
[constr_1323]	Applicability of attribute <code>ReceiverComSpec.usesEndToEndProtection</code>
[constr_1363]	Existence of attributes of <code>DiagnosticValueNeeds</code>
[constr_1364]	Existence of attributes of <code>DiagnosticIoControlNeeds</code>
[constr_1375]	Existence of attributes of <code>CompuMethod</code> and related meta-classes

**Table C.30: Added Constraints in R4.2.1**

### C.7.5 Changed Constraints in R4.2.1

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_1014]	Supported value encodings for <code>SwBaseType</code>
[constr_1015]	Prioritization of <code>SwDataDefProps</code>
[constr_1037]	Client shall not be connected to multiple servers
[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_1101]	Mode-related communication
[constr_1112]	Constraints of <code>dataIdMode</code> in <code>PROFILE_01</code>
[constr_1149]	<code>PortPrototypes</code> used for NV data management
[constr_1188]	Existence of <code>ReceiverComSpec.replaceWith</code>





Number	Heading
[constr_1288]	Allowed Attributes vs. <a href="#">category</a> for <a href="#">DataPrototypes</a> typed by <a href="#">Implementation-DataTypes</a>
[constr_1289]	Allowed Attributes vs. <a href="#">category</a> for <a href="#">DataPrototypes</a> typed by <a href="#">Application-DataTypes</a>
[constr_2009]	Supported kinds of <a href="#">PortPrototypes</a> of a <a href="#">NvBlockSwComponentType</a>
[constr_2015]	Limitation of <a href="#">SwcInternalBehavior</a> of a <a href="#">NvBlockSwComponentType</a>
[constr_2019]	<a href="#">ServiceSwComponentType</a> shall have service ports only
[constr_2022]	Mutually exclusive use of <a href="#">SynchronousServerCallPoints</a> and <a href="#">AsynchronousServerCallPoints</a>
[constr_2051]	Mandatory information of a <a href="#">SwValueCont</a>

**Table C.31: Changed Constraints in R4.2.1**

### C.7.6 Deleted Constraints in R4.2.1

Number	Heading
[constr_1189]	Allowed targets of <a href="#">externalReplacement</a>
[constr_1293]	Existence of <a href="#">DiagnosticEventNeeds.dtcNumber</a>
[constr_1294]	Existence of <a href="#">DiagnosticEventInfoNeeds.dtcNumber</a>
[constr_2551]	<a href="#">SwCalprmAxis.baseType</a> shall be ignored

**Table C.32: Deleted Constraints in R4.2.1**

## C.8 Constraint History of this Document according to AUTOSAR R4.2.2

### C.8.1 Added Traceables in R4.2.2

Number	Heading
[TPS_SWCT_01637]	Initial value for a specific <a href="#">implicitInterRunnableVariable</a> or <a href="#">explicitInterRunnableVariable</a>
[TPS_SWCT_01638]	Existence of <a href="#">SwConnector</a> between two <a href="#">PRPortPrototypes</a>
[TPS_SWCT_01639]	<a href="#">AtomicSwComponentType</a> offers a <a href="#">PPortPrototype</a> typed by <a href="#">ClientServerInterface</a> 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 <a href="#">PortInterface</a> for the Data Services
[TPS_SWCT_01641]	Definition of an “old-world” dynamic-size array data type by means of an <a href="#">ApplicationArrayDataType</a>





Number	Heading
[TPS_SWCT_01642]	Definition of an “old-world” dynamic-size array data type by means of an <a href="#">ImplementationDataType</a>
[TPS_SWCT_01644]	Definition of a “new-world” variable-size array data type by means of an <a href="#">ApplicationArrayDataType</a>
[TPS_SWCT_01645]	Definition of a “new-world” variable-size array data type by means of an <a href="#">ImplementationDataType</a>
[TPS_SWCT_01646]	Sending <a href="#">invalidValue</a> without invalidation applied by RTE/Com
[TPS_SWCT_01647]	<a href="#">Size Indicator</a> for <a href="#">dynamicArraySizeProfile</a> set to <a href="#">VSA_LINEAR</a> , <a href="#">VSA_SQUARE</a> , or <a href="#">VSA_FULLY_FLEXIBLE</a> if only <a href="#">ImplementationDataType</a> is present
[TPS_SWCT_01648]	<a href="#">Size Indicator</a> for <a href="#">dynamicArraySizeProfile</a> set to <a href="#">VSA_RECTANGULAR</a> if only <a href="#">ImplementationDataType</a> is present
[TPS_SWCT_01649]	Payload for <a href="#">dynamicArraySizeProfile</a> if only <a href="#">ImplementationDataType</a> is present
[TPS_SWCT_01650]	Structure of the <a href="#">VSA ImplementationDataType</a>
[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]	<a href="#">AtomicSwComponentType</a> offers <a href="#">PortPrototypes</a> typed by <a href="#">SenderReceiverInterfaces</a> to adjust the IO signal via diagnostic services
[TPS_SWCT_01655]	Reference from <a href="#">DiagnosticIoControlNeeds</a> to <a href="#">DiagnosticValueNeeds</a>
[TPS_SWCT_01656]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for <a href="#">IOControlRequest</a> and <a href="#">IOControlResponse</a>
[TPS_SWCT_01657]	NamingRule for <a href="#">RPortPrototype</a> referenced by a <a href="#">RoleBasedPortAssignment</a> with attribute <a href="#">role</a> set to “IOControlRequest”
[TPS_SWCT_01658]	NamingRule for <a href="#">PPortPrototype</a> referenced by a <a href="#">RoleBasedPortAssignment</a> with attribute <a href="#">role</a> set to “IOControlResponse”
[TPS_SWCT_01659]	Mapping of <a href="#">VariableDataPrototype</a> to a <a href="#">NvBlockDescriptor</a>

**Table C.33: Added Traceables in 4.2.2**

## C.8.2 Changed Traceables in R4.2.2

Number	Heading
[TPS_SWCT_01133]	<a href="#">AtomicSwComponentType</a> provides information about aging cycles
[TPS_SWCT_01158]	Three cases for <a href="#">PortInterfaceMapping</a>
[TPS_SWCT_01179]	Compound Primitive Data Type
[TPS_SWCT_01244]	Data types for calibration parameters are also described as primitive types
[TPS_SWCT_01370]	<a href="#">VariationPointProxy</a>





Number	Heading
[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 with 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>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>COM_AXIS</code> , <code>RES_AXIS</code> are for display only

**Table C.34: Changed Traceables in R4.2.2**

### C.8.3 Deleted Traceables in R4.2.2

Number	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 R4.2.2**

### C.8.4 Added Constraints in R4.2.2

Number	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 <code>category</code> <code>TYPE_REFERENCE</code>





Number	Heading
[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 R4.2.2**

### C.8.5 Changed Constraints in R4.2.2

Number	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>





Number	Heading
[constr_1060]	Compatibility of data types with <code>category</code> ARRAY, VAL_BLK
[constr_1064]	Compatibility of data types with <code>category</code> COM_AXIS, RES_AXIS, CURVE, MAP, CUBOID, CUBE_4, or CUBE_5
[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 VSA_LINEAR for <code>ImplementationDataType</code>
[constr_1319]	Profile VSA_SQUARE for <code>ImplementationDataType</code>
[constr_1320]	Profile VSA_RECTANGULAR for <code>ImplementationDataType</code>
[constr_1321]	Profile VSA_FULLY_FLEXIBLE for <code>ImplementationDataType</code>
[constr_1322]	<code>Size Indicator</code> for undefined <code>dynamicArraySizeProfile</code>
[constr_2051]	Mandatory information of a <code>SwValueCont</code>
[constr_2058]	Mandatory information of a <code>RuleBasedValueCont</code>

**Table C.37: Changed Constraints in R4.2.2**

### C.8.6 Deleted Constraints in R4.2.2

Number	Heading
[constr_1002]	End-to-end protection does not support n:1 communication
[constr_1067]	<code>ApplicationDataType</code> is or is not compatible to specific <code>Implementation-DataType</code>
[constr_2001]	Initial value for a specific <code>implicitInterRunnableVariable</code> or <code>explicitInterRunnableVariable</code>

**Table C.38: Deleted Constraints in R4.2.2**

## C.9 Constraint History of this Document according to AUTOSAR R4.3.0

### C.9.1 Added Traceables in R4.3.0

Number	Heading
[TPS_SWCT_01660]	Values of <code>SwcServiceDependency.category</code> reserved by the standard
[TPS_SWCT_01661]	Default value of <code>SwcServiceDependency.category</code>
[TPS_SWCT_01662]	Applicability of <code>DataTypeMappingSets</code> inside an <code>NvBlockSwComponent-Type</code>





Number	Heading
[TPS_SWCT_01663]	<a href="#">dataReadAccess</a> vs. <a href="#">dataReceivePointByValue</a> or <a href="#">dataReceivePointByArgument</a>
[TPS_SWCT_01664]	BswM acts as a mode requester towards an application mode manager
[TPS_SWCT_01665]	Usage of <a href="#">SwcModeSwitchEvent</a> for triggering a write procedure of <i>nv data</i>
[TPS_SWCT_01666]	Semantics of <a href="#">ModeSwitchEventTriggeredActivity.role</a>
[TPS_SWCT_01667]	Avoidance of overlapping of directly adjacent intervals within <a href="#">CompuMethods</a>
[TPS_SWCT_01668]	<a href="#">SecOc</a> Use Case: obtain the verification status of secure communication
[TPS_SWCT_01672]	<a href="#">SecOc</a> Use Case: software component retires from secure communication for a given period
[TPS_SWCT_01673]	Application Software Component sends requests using the J1939Rm
[TPS_SWCT_01674]	Application Software Component accepts requests using the J1939Rm
[TPS_SWCT_01675]	Recommendations for attributes of <a href="#">NvBlockNeeds</a> or for <a href="#">NvBlockDescriptor</a>
[TPS_SWCT_01676]	Preferred approach to checking the compatibility of input value and axis
[TPS_SWCT_01677]	Fall-back approach to checking the compatibility of input value and axis
[TPS_SWCT_01678]	StbM use Case: Application software component accesses the Synchronized Time-Base Manager
[TPS_SWCT_01679]	StbM use Case: Synchronized Time-Base Manager notifies application software component
[TPS_SWCT_01680]	Dem Use Case: Atomic Software-Component implements a Hardware Shutdown
[TPS_SWCT_01681]	Context path in <a href="#">ArVariableInImplementationDataInstanceRef</a>
[TPS_SWCT_01682]	The meaning of E2E-related attributes in a <a href="#">ReceiverComSpec</a> if a <a href="#">TransformationComSpecProps</a> of type <a href="#">EndToEndTransformationComSpecProps</a> is defined.
[TPS_SWCT_01683]	Specification of an array of input variable for an array of axes
[TPS_SWCT_01684]	Specification of a single input variable for an array of axes
[TPS_SWCT_01685]	Specification of an array of group axes for an array of <a href="#">category CURVE, MAP, CUBOID, CUBE_4, or CUBE_5</a>
[TPS_SWCT_01686]	Specification of a single group axis for an array of elements of <a href="#">category CURVE, MAP, CUBOID, CUBE_4, or CUBE_5</a>
[TPS_SWCT_01687]	Support of locked communication buffers
[TPS_SWCT_01688]	<a href="#">initValue</a> should exist in an <a href="#">RPortPrototype</a>
[TPS_SWCT_01689]	Relation between <a href="#">SwcServiceDependency</a> s and <a href="#">PortPrototypes</a>
[TPS_SWCT_01690]	<a href="#">AtomicSwComponentType</a> offers a <a href="#">PPortPrototype</a> typed by <a href="#">ClientServerInterface</a> to read/write and IOCtrl current value via diagnostic services
[TPS_SWCT_01691]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Data Services
[TPS_SWCT_01692]	Meaning of <a href="#">CompositeRuleBasedValueSpecification</a>
[TPS_SWCT_01693]	Usage of <a href="#">VendorSpecificServiceNeeds</a>





Number	Heading
[TPS_SWCT_01694]	Setup for Default Error Tracer Service use Case: development errors or run-time error
[TPS_SWCT_01695]	Relation between <i>ValueSpecification</i> and the definition of <i>CompuScales</i>
[TPS_SWCT_01696]	<i>CompuScale</i> Value Symbolic Name
[TPS_SWCT_01697]	Supported development approach for software-components that interact with AUTOSAR services
[TPS_SWCT_01698]	Attributes that are subject to development approach
[TPS_SWCT_01699]	Usage of <i>ApplicationArrayElement.indexDataType</i>
[TPS_SWCT_01700]	Definition of unions that can be transmitted over a communication network
[TPS_SWCT_01701]	Wrapped Union Data Type
[TPS_SWCT_01702]	Initialization of the “payload” of a <i>Wrapped Union Data Type</i>
[TPS_SWCT_01703]	Setup for <i>AtomicSwComponentType</i> which sets or gets the Global Supervision Status
[TPS_SWCT_01704]	Definition of supervised entity
[TPS_SWCT_01705]	Definition of Checkpoints
[TPS_SWCT_01706]	<i>AtomicSwComponentType</i> supports <i>DiagnosticSessionControl</i> to get informed about the diagnostic session
[TPS_SWCT_01707]	<i>AtomicSwComponentType</i> supports <i>EcuReset</i> service via diagnostic services
[TPS_SWCT_01708]	<i>AtomicSwComponentType</i> supports <i>EcuReset ModeRapidPowerShutDown</i> service via diagnostic services
[TPS_SWCT_01709]	<i>AtomicSwComponentType</i> supports <i>CommunicationControl</i> service via diagnostic services
[TPS_SWCT_01710]	Suffix used for the resulting name of the <i>PortInterface</i> for the Communication Control
[TPS_SWCT_01711]	<i>AtomicSwComponentType</i> supports <i>ControlDTCSetting</i> service via diagnostic services
[TPS_SWCT_01712]	<i>AtomicSwComponentType</i> supports <i>SecurityAccess</i> to get informed about the security level
[TPS_SWCT_01713]	<i>ExclusiveArea</i> is entered and exited by a common set of APIs
[TPS_SWCT_01714]	<i>ExclusiveArea</i> is entered and exited by an individual set of APIs
[TPS_SWCT_01715]	Software-Component wants to be triggered on Monitor Status Changes
[TPS_SWCT_01716]	<i>SecOc</i> Use Case: deliver freshness to SecOC I
[TPS_SWCT_01717]	<i>SecOc</i> Use Case: deliver freshness to SecOC II
[TPS_SWCT_01718]	<i>SecOc</i> Use Case: deliver freshness to SecOC III
[TPS_SWCT_01719]	Selection of applicable <i>RptProfiles</i>
[TPS_SWCT_01720]	Preparation Levels
[TPS_SWCT_01721]	References from <i>RptContainer</i>
[TPS_SWCT_01722]	Semantics of <i>RP Service Point</i>
[TPS_SWCT_01723]	Semantics of <i>RP Service Function</i>
[TPS_SWCT_01724]	Semantics of <i>RapidPrototypingScenario</i>
[TPS_SWCT_01725]	<i>AtomicSwComponentType</i> uses the secure counter of the Crypto Service







Number	Heading
[TPS_SWCT_01726]	<code>AtomicSwComponentType</code> uses the key management of the Crypto Service
[TPS_SWCT_01727]	Suffix used for the resulting name of the <code>PortInterface</code> for crypto <code>PortInterfaces</code>
[TPS_SWCT_01728]	V2xFac Use Case: Application software component provides Vehicle specific data to the V2X-Stack for CAM transmission
[TPS_SWCT_01729]	V2xFac Use Case: V2xFac notifies application software component about received messages
[TPS_SWCT_01730]	V2xFac Use Case: Application software component triggers transmission of DENM message
[TPS_SWCT_01731]	V2xM Use Case: Application software component provides Vehicle specific data to the V2X-Stack for Position and Time information
[TPS_SWCT_01732]	V2xM Use Case: Application software component needs V2X specific data from the V2X Manager
[TPS_SWCT_01733]	V2xM Use Case: Application software component has soft-control over Pseudonym-Change within V2X Manager
[TPS_SWCT_01734]	V2xM Use Case: Application software component has the ability to do Verification-on-Demand
[TPS_SWCT_01735]	V2xM Use Case: Application software component do location based calculations

**Table C.39: Added Traceables in 4.3.0**

### C.9.2 Changed Traceables in R4.3.0

Number	Heading
[TPS_SWCT_01005]	Usage of <code>SwcServiceDependencies</code> for vendor-specific services
[TPS_SWCT_01054]	Semantics of the <code>explicitInterRunnableVariable</code>
[TPS_SWCT_01134]	<code>AtomicSwComponentType</code> enables reporting of DTCs in general
[TPS_SWCT_01158]	Cases for <code>PortInterfaceMapping</code>
[TPS_SWCT_01173]	Control port shall <b>not</b> become a part of the <code>PortGroup</code>
[TPS_SWCT_01174]	Status port shall <b>not</b> become a member of the <code>PortGroup</code>
[TPS_SWCT_01182]	Conceptual levels for the definition of initial values
[TPS_SWCT_01275]	values of the attribute <code>swImplPolicy</code> are restricted depending on the context
[TPS_SWCT_01331]	<code>dataWriteAccess</code> vs. <code>dataSendPoint</code>
[TPS_SWCT_01355]	<code>enableTakeAddress</code> = true
[TPS_SWCT_01463]	<code>ModeDeclarationGroupPrototypeMapping.modeDeclarationMappingSet</code> defines the applicable set of <code>ModeDeclarationMappings</code>
[TPS_SWCT_01493]	The number of <code>RuleBasedValueSpecification.arguments</code> shall not exceed the array size
[TPS_SWCT_01494]	A <code>RuleBasedValueSpecification</code> of rule <code>FILL_UNTIL_END</code> shall fill the value of the last <code>RuleBasedValueSpecification.arguments</code> until the last element of the array



△

Number	Heading
[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_01569]	Definition of CompuScale Code Symbolic Name
[TPS_SWCT_01586]	Writing strategies for <i>nv data</i>
[TPS_SWCT_01609]	A <code>RuleBasedValueSpecification</code> of rule <code>FILL_UNTIL_MAX_SIZE</code> shall fill the value of the last <code>RuleBasedValueSpecification.arguments</code> until the number of elements specified in <code>maxSizeToFill</code>
[TPS_SWCT_01635]	Naming conventions may support the effectiveness of <code>SymbolProps</code>
[TPS_SWCT_02001]	Values of <code>SwAxisCont</code> with the <code>category COM_AXIS, RES_AXIS</code> are for display only
[TPS_SWCT_02014]	<code>AtomicSwComponentType</code> supports Response On Event (ROE) via diagnostic services
[TPS_SWCT_02020]	<code>AtomicSwComponentType</code> uses the hash calculation of the Crypto Service
[TPS_SWCT_02021]	<code>AtomicSwComponentType</code> uses the message authentication code (MAC) calculation of the Crypto Service
[TPS_SWCT_02022]	<code>AtomicSwComponentType</code> uses the message authentication code (MAC) verification of the Crypto Service
[TPS_SWCT_02024]	<code>AtomicSwComponentType</code> uses the generation of random numbers of the Crypto Service
[TPS_SWCT_02025]	<code>AtomicSwComponentType</code> uses the Authenticated Encryption with Associated Data (AEAD) of the Crypto Service
[TPS_SWCT_02026]	<code>AtomicSwComponentType</code> uses the Authenticated Encryption with Associated Data (AEAD) of the Crypto Service
[TPS_SWCT_02027]	<code>AtomicSwComponentType</code> uses the encryption of the Crypto Service
[TPS_SWCT_02028]	<code>AtomicSwComponentType</code> uses the decryption of the Crypto Service
[TPS_SWCT_02031]	<code>AtomicSwComponentType</code> uses the signature generation of the Crypto Service
[TPS_SWCT_02032]	<code>AtomicSwComponentType</code> uses the signature verification of the Crypto Service
[TPS_SWCT_02053]	Values of <code>RuleBasedAxisCont</code> with the <code>category COM_AXIS, RES_AXIS</code> are for display only
[TPS_SWCT_02503]	Setup for NVM Use Case: Software-Components using Nv Data provided by <code>NvBlockSwComponentType</code>
[TPS_SWCT_02506]	Setup for Dlt use Case: Application software component accesses the Dlt module

**Table C.40: Changed Traceables in R4.3.0**

### C.9.3 Deleted Traceables in R4.3.0

Number	Heading
[TPS_SWCT_01279]	Preferred conversion direction depends on the use case
[TPS_SWCT_01428]	<a href="#">ServiceSwComponentType</a> representing the <code>Dem</code> provides a <a href="#">PPortPrototype</a> for the <code>Dcm</code>
[TPS_SWCT_02018]	Setup for <a href="#">AtomicSwComponentType</a> which contains a Supervised Entity
[TPS_SWCT_02023]	<a href="#">AtomicSwComponentType</a> uses the generation of random seed of the Crypto Service
[TPS_SWCT_02029]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical encryption of the Crypto Service
[TPS_SWCT_02030]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical decryption of the Crypto Service
[TPS_SWCT_02033]	<a href="#">AtomicSwComponentType</a> uses the checksum calculation of the Crypto Service
[TPS_SWCT_02034]	<a href="#">AtomicSwComponentType</a> uses the key derivation of the Crypto Service
[TPS_SWCT_02035]	<a href="#">AtomicSwComponentType</a> uses the symmetric key derivation of the Crypto Service
[TPS_SWCT_02036]	<a href="#">AtomicSwComponentType</a> uses the key exchange interface for public value calculation of the Crypto Service
[TPS_SWCT_02037]	<a href="#">AtomicSwComponentType</a> uses the key exchange interface for secret value calculation of the Crypto Service
[TPS_SWCT_02038]	<a href="#">AtomicSwComponentType</a> uses the key exchange interface to calculate symmetric key with the Crypto Service
[TPS_SWCT_02039]	<a href="#">AtomicSwComponentType</a> uses the symmetrical key extraction of the Crypto Service
[TPS_SWCT_02040]	<a href="#">AtomicSwComponentType</a> uses the symmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a symmetric key
[TPS_SWCT_02041]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a asymmetric key
[TPS_SWCT_02042]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical public key extraction of the Crypto Service
[TPS_SWCT_02043]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical private key extraction of the Crypto Service
[TPS_SWCT_02044]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a symmetrical wrapping key
[TPS_SWCT_02045]	<a href="#">AtomicSwComponentType</a> uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a asymmetrical wrapping key

**Table C.41: Deleted Traceables in R4.3.0**

### C.9.4 Added Constraints in R4.3.0

Number	Heading
[constr_1407]	Definition of <code>SwDataDefProps.dataConstr</code> depending on the capabilities of the data type
[constr_1408]	Definition of <code>SwDataDefProps.displayFormat</code> depending on the capabilities of the data type
[constr_1409]	Definition of <code>SwDataDefProps.dataConstr</code> depending on the capabilities of the element data type
[constr_1410]	Definition of <code>SwDataDefProps.displayFormat</code> depending on the capabilities of the element data type
[constr_1413]	Definition of <code>SwDataDefProps.stepSize</code> depending on the capabilities of the data type
[constr_1414]	Definition of <code>SwDataDefProps.stepSize</code> depending on the capabilities of the element data type
[constr_1415]	Supported values of <code>ModeSwitchEventTriggeredActivity.role</code>
[constr_1416]	Existence of <code>ApplicationArrayElement.maxNumberOfElements</code>
[constr_1417]	Invalid connection between <code>NvBlockSwComponentType</code> and other <code>AtomicSwComponentType</code> (I)
[constr_1418]	Invalid connection between <code>NvBlockSwComponentType</code> and other <code>AtomicSwComponentType</code> (II)
[constr_1420]	Existence of <code>SwAxisIndividual.inputVariableType</code>
[constr_1422]	Value of <code>category</code> is VOID
[constr_1423]	Completeness of references <code>ArVariableInImplementationDataInstanceRef.contextDataPrototype</code>
[constr_1424]	Existence of <code>ArVariableInImplementationDataInstanceRef.contextDataPrototype</code>
[constr_1425]	Definition of <code>swCalprmAxisSet.swCalprmAxis/ SwAxisIndividual.swVariableRef</code> depending on the capabilities of the data type
[constr_1426]	Consistency of array sizes for axes and input variable array
[constr_1427]	Definition of <code>swCalprmAxisSet.swCalprmAxis/ SwAxisGrouped.swCalprmRef</code> depending on the capabilities of the data type
[constr_1428]	Consistency of array sizes for arrays of elements of <code>category CURVE, MAP, CUBOID, CUBE_4, or CUBE_5</code> arrays and used group axes arrays
[constr_1429]	Access to data within <code>PortPrototypes</code> from within <code>RunnableEntitys</code>
[constr_1430]	Access to local data from within <code>RunnableEntitys</code>
[constr_1431]	Access to parameters from within <code>RunnableEntitys</code>
[constr_1432]	Multiplicity of <code>CommunicationBufferLocking</code>
[constr_1433]	Transient faults are not applicable to software-components
[constr_1434]	<code>CompuScale</code> s shall not have identical <code>CompuScale Value Symbolic Names</code>
[constr_1436]	<code>DiagnosticCommunicationManagerNeeds.serviceRequestCallbackType</code> is set to <code>requestCallbackTypeSupplier</code>
[constr_1437]	<code>DiagnosticCommunicationManagerNeeds.serviceRequestCallbackType</code> is set to <code>requestCallbackTypeManufacturer</code>





Number	Heading
[constr_1438]	<code>ApplicationArrayElement.indexDataType</code> needs to refer to a <code>CompuMethod</code> of category <code>TEXTTABLE</code>
[constr_1439]	Requirements on <code>ApplicationArrayElement</code> if attribute <code>indexDataType</code> exists
[constr_1440]	Size of the <code>CompuMethod</code> of category <code>TEXTTABLE</code> referenced by <code>ApplicationArrayElement.indexDataType</code>
[constr_1442]	category <code>TYPE_REFERENCE</code> shall not be used for modeling the “payload” of a <code>Wrapped Union Data Type</code>
[constr_1443]	category <code>UNION</code> shall not be used for <code>ImplementationDataType</code>
[constr_1444]	Limited applicability of <code>Wrapped Union Data Type</code>
[constr_1445]	Initialization of the <code>Member Selector</code> of a <code>Wrapped Union Data Type</code>
[constr_1446]	No definition of <code>invalidValue</code> for a <code>Wrapped Union Data Type</code>
[constr_1468]	Limitation on the number of <code>SwcExclusiveAreaPolicys</code>
[constr_1469]	Applicability of constraints depending on the existence of a data transformation

**Table C.42: Added Constraints in R4.3.0**

### C.9.5 Changed Constraints in R4.3.0

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_1011]	category of <code>SwBaseType</code>
[constr_1014]	Supported value encodings for <code>SwBaseType</code>
[constr_1015]	Prioritization of <code>SwDataDefProps</code>
[constr_1024]	Stepwise definition of <code>CompuMethods</code>
[constr_1045]	Supported value encodings for <code>SwBaseType</code> in the context of <code>PortInterfaces</code>
[constr_1105]	Value of <code>arraySize</code>
[constr_1240]	Consistency of <code>ArgumentDataPrototypes</code> within the context of a <code>ClientServer-OperationMapping</code>
[constr_1271]	<code>RecordValueSpecification.fields</code> shall be identical to the number of <code>ApplicationRecordDataType.elements</code>
[constr_1272]	<code>RecordValueSpecification.fields</code> shall be identical to the number of <code>subElements</code> of <code>ImplementationDataType</code> of category <code>STRUCTURE</code>
[constr_1284]	Limitation of the use of <code>TextValueSpecification</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_1302]	Restriction of data invalidation





Number	Heading
[constr_1375]	Existence of attributes of <code>CompuMethod</code> and related meta-classes
[constr_1400]	Reference to a specific <code>DataTransformation</code>
[constr_2006]	Number of <code>AsynchronousServerCallResultPoint</code> referencing to one <code>AsynchronousServerCallPoint</code>
[constr_2015]	Limitation of <code>SwcInternalBehavior</code> of a <code>NvBlockSwComponentType</code>
[constr_2041]	<code>swImplPolicy</code> for <code>VariableDataPrototype</code> in the role <code>staticMemory</code>
[constr_2051]	Mandatory information of a <code>SwValueCont</code>
[constr_2058]	Mandatory information of a <code>RuleBasedValueCont</code>

**Table C.43: Changed Constraints in R4.3.0**

### C.9.6 Deleted Constraints in R4.3.0

Number	Heading
[constr_1019]	Compatibility of input value and axis
[constr_1021]	A <code>CompuMethod</code> shall specify instructions for both directions
[constr_1133]	Identical <code>CompuScale</code> <code>Symbolic</code> <code>Names</code> shall have the same range
[constr_1201]	<code>initValue</code> shall exist in an <code>RPortPrototype</code>
[constr_1323]	Applicability of attribute <code>ReceiverComSpec.usesEndToEndProtection</code>

**Table C.44: Deleted Constraints in R4.3.0**

## C.10 Constraint History of this Document according to AUTOSAR R4.3.1

### C.10.1 Added Traceables in R4.3.1

Number	Heading
[TPS_SWCT_01736]	Default values for <code>Unit.physicalDimension</code>
[TPS_SWCT_01737]	Default values for physical exponents
[TPS_SWCT_01738]	Context path in <code>ArParameterInImplementationDataInstanceRef</code>
[TPS_SWCT_01739]	Function Inhibition Manager Use Case: react on suppressed or unavailable events
[TPS_SWCT_01740]	StbM use Case: Process time snapshot obtained from global time slave for diagnostics purposes
[TPS_SWCT_01741]	Suffix used for the resulting name of the <code>PortInterface</code> for measurement notification
[TPS_SWCT_01742]	StbM use Case: Software-component represents a global time master





Number	Heading
[TPS_SWCT_01743]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the global time master role
[TPS_SWCT_01744]	StbM use Case: Software-component represents a global time slave
[TPS_SWCT_01745]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the global time slave role
[TPS_SWCT_01746]	Atomic Software-Component provides the further action byte to the DoIP Service Component
[TPS_SWCT_01747]	Value of <a href="#">category</a> for fix axis
[TPS_SWCT_01748]	Sub-categories of fix axes
[TPS_SWCT_01749]	Semantics of <a href="#">SwAxisGeneric.swAxisType</a> in the definition of a fix axis
[TPS_SWCT_01750]	Semantics of <a href="#">SwAxisGeneric.swGenericAxisParam</a> in the definition of a fix axis

**Table C.45: Added Traceables in R4.3.1**

### C.10.2 Changed Traceables in R4.3.1

Number	Heading
[TPS_SWCT_01193]	Mappings between application and implementation types do not necessarily have to form a 1:1 relation
[TPS_SWCT_01222]	Applicability of <a href="#">DataFilter</a>
[TPS_SWCT_01225]	<a href="#">RunnableEntity</a> implements the functionality of more than one <a href="#">ClientServerOperations</a>
[TPS_SWCT_01288]	Interpretation of <a href="#">PhysConstrs</a> and <a href="#">InternalConstrs</a> by tools
[TPS_SWCT_01444]	Size of <a href="#">SwBaseType</a> is specified in bits
[TPS_SWCT_01560]	Supported <a href="#">categorys</a> of <a href="#">CompuMethods</a> for data conversion
[TPS_SWCT_01675]	Recommendations for attributes of <a href="#">NvBlockNeeds</a> or for <a href="#">NvBlockDescriptor</a>
[TPS_SWCT_01678]	StbM use Case: start timer and potentially get notified about its expiration
[TPS_SWCT_01679]	StbM use Case: Software-Components wants to get notifications of status changes
[TPS_SWCT_01714]	<a href="#">ExclusiveArea</a> is entered and exited by an individual set of APIs
[TPS_SWCT_02506]	Setup for Dlt use Case: Application software component accesses the Dlt module

**Table C.46: Changed Traceables in R4.3.1**

### C.10.3 Deleted Traceables in R4.3.1

Number	Heading
[TPS_SWCT_01490]	AUTOSAR supports <code>ApplicationErrors</code> only for <code>ClientServerInterfaces</code>

**Table C.47: Deleted Traceables in R4.3.1**

### C.10.4 Added Constraints in R4.3.1

Number	Heading
[constr_1515]	Existence of <code>ImplementationDataTypeSubElementRef.implementationDataTypeElement</code> as opposed to <code>ImplementationDataTypeSubElementRef.parameterImplementationDataTypeElement</code>
[constr_1516]	Completeness of references <code>ArParameterInImplementationDataInstanceRef.contextDataPrototype</code>
[constr_1517]	Existence of <code>ArParameterInImplementationDataInstanceRef.contextDataPrototype</code>
[constr_1518]	Consistency of data types in the context of <code>ArParameterInImplementationDataInstanceRef</code>
[constr_1519]	Existence of attributes vs. <code>category</code> of <code>ApplicationValueSpecification</code>
[constr_1520]	Semantics of <code>ObdRatioServiceNeeds.rateBasedMonitoredEvent</code>
[constr_1521]	Reference from <code>AsynchronousServerCallReturnsEvent</code> to <code>AsynchronousServerCallResultPoint</code>
[constr_1523]	No mode disabling for <code>OperationInvokedEvents</code>
[constr_1538]	Restriction for <code>ReceiverComSpec.dataElement</code>
[constr_1539]	Restriction for <code>SenderComSpec.dataElement</code>
[constr_1540]	Existence of <code>ClientComSpec.operation</code>
[constr_1541]	Existence of <code>ServerComSpec.operation</code>
[constr_1544]	Modeling of <code>SwAxisGeneric</code> for the definition of a fix axis
[constr_1545]	No initialization for fix axis

**Table C.48: Added Constraints in R4.3.1**

### C.10.5 Changed Constraints in R4.3.1

Number	Heading
[constr_1004]	Mapping of <code>ApplicationDataTypes</code> in the scope of single <code>AtomicSwComponentTypes</code>
[constr_1007]	Allowed attributes of <code>SwDataDefProps</code> for <code>ApplicationDataTypes</code>
[constr_1009]	<code>SwDataDefProps</code> applicable to <code>ImplementationDataTypes</code>





△

Number	Heading
[constr_1011]	<code>category</code> of <code>SwBaseType</code>
[constr_1012]	Value of <code>category</code> is <code>FIXED_LENGTH</code>
[constr_1015]	Prioritization of <code>SwDataDefProps</code>
[constr_1038]	Reference to <code>ApplicationError</code>
[constr_1044]	Applicability of <code>DataFilter</code>
[constr_1090]	<code>WaitPoint</code> and <code>RunnableEntity</code>
[constr_1102]	<code>ApplicationError</code> in the scope of one <code>SwComponentType</code>
[constr_1126]	Compatibility of <code>DataConstrs</code>
[constr_1220]	Compatibility of <code>SwBaseType</code>
[constr_1229]	<code>category</code> of <code>ImplementationDataType</code> boils down to <code>VALUE</code>
[constr_1288]	Allowed Attributes vs. <code>category</code> for <code>DataPrototypes</code> typed by <code>ImplementationDataTypes</code>
[constr_1289]	Allowed Attributes vs. <code>category</code> for <code>DataPrototypes</code> typed by <code>ApplicationDataTypes</code>
[constr_1422]	Value of <code>category</code> is <code>VOID</code>
[constr_2027]	<code>SwcServiceDependency</code> shall be defined for service ports only
[constr_2043]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in the role <code>romBlock</code>
[constr_2044]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in the role <code>sharedParameter</code>
[constr_2045]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in the role <code>perInstanceParameter</code>
[constr_2046]	<code>swImplPolicy</code> for <code>ParameterDataPrototype</code> in the role <code>constantMemory</code>

**Table C.49: Changed Constraints in R4.3.1**

### C.10.6 Deleted Constraints in R4.3.1

Number	Heading
[constr_1013]	Value of <code>category</code> is <code>VARIABLE_LENGTH</code>
[constr_1436]	<code>DiagnosticCommunicationManagerNeeds.serviceRequestCallbackType</code> is set to <code>requestCallbackTypeSupplier</code>
[constr_1437]	<code>DiagnosticCommunicationManagerNeeds.serviceRequestCallbackType</code> is set to <code>requestCallbackTypeManufacturer</code>

**Table C.50: Deleted Constraints in R4.3.1**

## C.11 Constraint History of this Document according to AUTOSAR R4.4.0

### C.11.1 Added Traceables in R4.4.0

Number	Heading
[TPS_SWCT_01751]	The meaning of E2E-related attributes in a <code>SenderComSpec</code> if a <code>TransformationComSpecProps</code> of type <code>EndToEndTransformationComSpecProps</code> is defined
[TPS_SWCT_01752]	Initialization of a variable-size array
[TPS_SWCT_01753]	Application of compatibility rules for <code>ReceiverComSpec.replaceWith</code>
[TPS_SWCT_01754]	<code>initValue</code> defined in the context of a <code>ComSpec</code>
[TPS_SWCT_01755]	Duplicate existence of <code>initValue</code> in the context of a <code>PRPortPrototype</code> typed by an <code>NvDataInterface</code>
[TPS_SWCT_01756]	Semantics of <code>SwDataDefProps.displayPresentation</code>
[TPS_SWCT_01757]	Not-applicable scenario for <code>presentationContinuous</code>
[TPS_SWCT_01758]	Applicable value range of <code>SwDataDefProps.displayPresentation</code>
[TPS_SWCT_01759]	Use cases for unions
[TPS_SWCT_01760]	Defining the dimension of an <code>ApplicationPrimitiveDataType</code> of category <code>VAL_BLK</code>
[TPS_SWCT_01761]	Physical limits of pure textual conversions
[TPS_SWCT_01762]	Physical limits of mixed textual conversions
[TPS_SWCT_01763]	HtssM Service Use Case: Query results of hardware tests
[TPS_SWCT_01764]	V2xFac Use Case: Application software component shall be able to process the <i>MAP (topology) Extended Message</i>
[TPS_SWCT_01765]	Dem Service Use Case: In-Use-Monitoring Performance Ratio Denominator interface
[TPS_SWCT_01766]	Software-component computes the PIDs, and pushes them to Dem for storage and reporting to Dcm
[TPS_SWCT_01767]	Software-component located on an OBD master ECU reads the PID 21, and and sends the value around via “regular” sender-receiver communication
[TPS_SWCT_01768]	Semantics of <code>DataPrototypeMapping.secondToFirstDataTransformation</code>
[TPS_SWCT_01769]	Dcm Use Case: Upload and download of data
[TPS_SWCT_01770]	V2xFac Use Case: Application software component processes Infrastructure to Vehicle Information Message
[TPS_SWCT_01771]	Definition of optional elements on the level of <code>ApplicationDataType</code>
[TPS_SWCT_01772]	Semantics of attribute <code>ImplementationDataType.isStructWithOptionalElement</code>
[TPS_SWCT_01773]	Definition of <code>Optional Element Structure</code> on the level of <code>ImplementationDataType</code>
[TPS_SWCT_01774]	Modeling of <code>ImplementationDataType</code> with optional elements
[TPS_SWCT_01775]	Structured data types with optional elements





Number	Heading
[TPS_SWCT_01776]	<i>AtomicSwComponentType</i> uses the API of the Crypto Service to set a key valid
[TPS_SWCT_01777]	<i>AtomicSwComponentType</i> uses the API of the Crypto Service to create a random seed
[TPS_SWCT_01778]	<i>AtomicSwComponentType</i> uses the API of the Crypto Service to generate a key
[TPS_SWCT_01779]	<i>AtomicSwComponentType</i> uses the API of the Crypto Service to derive a key
[TPS_SWCT_01780]	<i>AtomicSwComponentType</i> uses the API of the Crypto Service to execute calculation of the public value for key exchange
[TPS_SWCT_01781]	<i>AtomicSwComponentType</i> uses the API of the Crypto Service to execute calculation of shared secret for key exchange
[TPS_SWCT_01782]	<i>AtomicSwComponentType</i> uses the API of the Crypto Service to execute certificate parsing
[TPS_SWCT_01783]	<i>AtomicSwComponentType</i> uses the API of the Crypto Service to execute certificate verification
[TPS_SWCT_01784]	<i>SecOc</i> Use Case: enable the sending of Pdus even if computation of the MAC is not possible
[TPS_SWCT_01785]	Initial value for <i>ImplementationDataType</i> of category <i>STRUCTURE</i> where attribute <i>isStructWithOptionalElement</i> set to the value <i>True</i>
[TPS_SWCT_01786]	Initial value for the <i>ImplementationDataTypeElement</i> where the <i>short-Name</i> is set to the value <i>availabilityBitfield</i>
[TPS_SWCT_01787]	Initialization of not-available <i>ImplementationDataTypeElement</i>
[TPS_SWCT_01788]	V2xFac Use Case: Application software component processes Signal Phase And Timing Extended Message
[TPS_SWCT_01789]	<i>AtomicSwComponentType</i> implements a Diagnostic Monitor that provides monitor data, debouncing by Dem
[TPS_SWCT_01790]	<i>AtomicSwComponentType</i> implements a Diagnostic Monitor that provides monitor data, debouncing by software-component
[TPS_SWCT_01791]	<i>AtomicSwComponentType</i> acts as a "file server" to a diagnostic tester

**Table C.51: Added Traceables in R4.4.0**

### C.11.2 Changed Traceables in R4.4.0

Number	Heading
[TPS_SWCT_01016]	<i>AtomicSwComponentType</i> wants to select a shutdown target
[TPS_SWCT_01017]	<i>AtomicSwComponentType</i> wants to select a boot target
[TPS_SWCT_01018]	<i>AtomicSwComponentType</i> wants to use an alarm clock
[TPS_SWCT_01028]	<i>AtomicSwComponentType</i> implements a Diagnostic Monitor
[TPS_SWCT_01029]	<i>AtomicSwComponentType</i> implements a Diagnostic Monitor





Number	Heading
[TPS_SWCT_01132]	<i>AtomicSwComponentType</i> provides information about operating cycles
[TPS_SWCT_01136]	<i>AtomicSwComponentType</i> retrieves information of the lamp status
[TPS_SWCT_01180]	Maximum possible size of Compound Primitive Data Type
[TPS_SWCT_01275]	values of the attribute <i>swImplPolicy</i> are restricted depending on the context
[TPS_SWCT_01374]	Implementation of <i>AutosarParameterRef</i>
[TPS_SWCT_01375]	Implementation of <i>AutosarVariableRef</i>
[TPS_SWCT_01398]	Communication patterns for AUTOSAR services
[TPS_SWCT_01405]	Creation of the <i>ServiceSwComponentTypes</i>
[TPS_SWCT_01426]	<i>AtomicSwComponentType</i> provides callback if any diagnostic event data and/or status changed
[TPS_SWCT_01456]	Predefined values for <i>MemorySection.option</i> and <i>SwAddrMethod.option</i>
[TPS_SWCT_01678]	StbM use Case: start timer and potentially get notified about its expiration
[TPS_SWCT_01698]	Attributes that are subject to development approach
[TPS_SWCT_01706]	<i>AtomicSwComponentType</i> supports <i>DiagnosticSessionControl</i> to get informed about the diagnostic session
[TPS_SWCT_01707]	<i>AtomicSwComponentType</i> supports <i>EcuReset</i> service via diagnostic services
[TPS_SWCT_01708]	<i>AtomicSwComponentType</i> supports <i>EcuReset ModeRapidPowerShutdown</i> service via diagnostic services
[TPS_SWCT_01709]	<i>AtomicSwComponentType</i> supports <i>CommunicationControl</i> service via diagnostic services
[TPS_SWCT_01711]	<i>AtomicSwComponentType</i> supports <i>ControlDTCSetting</i> service via diagnostic services
[TPS_SWCT_01712]	<i>AtomicSwComponentType</i> supports <i>SecurityAccess</i> to get informed about the security level
[TPS_SWCT_01715]	Software-Component wants to be triggered on Monitor Status Changes
[TPS_SWCT_01727]	Suffix used for the resulting name of the <i>PortInterface</i> for crypto <i>PortInterfaces</i>
[TPS_SWCT_01728]	V2xFac Use Case: Application software component provides Vehicle specific data to the V2X-Stack for CAM transmission
[TPS_SWCT_01729]	V2xFac Use Case: V2xFac notifies application software component about received messages
[TPS_SWCT_01730]	V2xFac Use Case: Application software component triggers transmission of DENM message
[TPS_SWCT_02000]	Default value for attribute <i>swImplPolicy</i>
[TPS_SWCT_02014]	<i>AtomicSwComponentType</i> supports Response On Event (ROE) via diagnostic services
[TPS_SWCT_02020]	<i>AtomicSwComponentType</i> uses the hash calculation of the Crypto Service
[TPS_SWCT_02021]	<i>AtomicSwComponentType</i> uses the message authentication code (MAC) calculation of the Crypto Service





Number	Heading
[TPS_SWCT_02022]	<a href="#">AtomicSwComponentType</a> uses the message authentication code (MAC) verification of the Crypto Service
[TPS_SWCT_02024]	<a href="#">AtomicSwComponentType</a> uses the generation of random numbers of the Crypto Service
[TPS_SWCT_02025]	<a href="#">AtomicSwComponentType</a> uses the Authenticated Encryption with Associated Data (AEAD) of the Crypto Service
[TPS_SWCT_02026]	<a href="#">AtomicSwComponentType</a> uses the Authenticated Encryption with Associated Data (AEAD) of the Crypto Service
[TPS_SWCT_02027]	<a href="#">AtomicSwComponentType</a> uses the encryption of the Crypto Service
[TPS_SWCT_02028]	<a href="#">AtomicSwComponentType</a> uses the decryption of the Crypto Service
[TPS_SWCT_02031]	<a href="#">AtomicSwComponentType</a> uses the signature generation of the Crypto Service
[TPS_SWCT_02032]	<a href="#">AtomicSwComponentType</a> uses the signature verification of the Crypto Service
[TPS_SWCT_02506]	Setup for Dlt use Case: Application software component accesses the Dlt module

**Table C.52: Changed Traceables in R4.4.0**

### C.11.3 Deleted Traceables in R4.4.0

Number	Heading
[TPS_SWCT_01012]	<a href="#">AtomicSwComponentType</a> reads the current ECU mode (fixed variant)
[TPS_SWCT_01013]	<a href="#">AtomicSwComponentType</a> shall keep the ECU alive (fixed variant)
[TPS_SWCT_01014]	<a href="#">AtomicSwComponentType</a> wants to select a shutdown target (fixed variant)
[TPS_SWCT_01015]	<a href="#">AtomicSwComponentType</a> wants to select a boot target (fixed variant)
[TPS_SWCT_01125]	<a href="#">serverArgumentImplPolicy</a>
[TPS_SWCT_01133]	<a href="#">AtomicSwComponentType</a> provides information about aging cycles
[TPS_SWCT_01658]	NamingRule for <a href="#">PPortPrototype</a> referenced by a <a href="#">RoleBasedPortAssignment</a> with attribute <code>role</code> set to "IOControlResponse"
[TPS_SWCT_01710]	Suffix used for the resulting name of the <a href="#">PortInterface</a> for the Communication Control
[TPS_SWCT_01725]	<a href="#">AtomicSwComponentType</a> uses the secure counter of the Crypto Service

**Table C.53: Deleted Traceables in R4.4.0**

### C.11.4 Added Constraints in R4.4.0

Number	Heading
[constr_1583]	PortInterfaceMapping for DataPrototype typed by Compound Primitive Data Type
[constr_1592]	Definition of SwDataDefProps.displayPresentation depending on the capabilities of the data type
[constr_1602]	Definition of SwDataDefProps.displayPresentation depending on the capabilities of the element
[constr_1607]	Only Wrapped Union Data Types in PortInterface
[constr_1608]	Existence of rootParameterDataPrototype
[constr_1609]	Existence of rootVariableDataPrototype
[constr_1610]	Existence of SwDataDefProps.swValueBlockSize and SwDataDefProps.swValueBlockSizeMult
[constr_1611]	Existence of ImplementationDataTypeSubElementRef.implementationDataTypeElement as opposed to ImplementationDataTypeSubElementRef.parameterImplementationDataTypeElement
[constr_1622]	Value of TimingEvent.offset vs. TimingEvent.period
[constr_1631]	Applicability of DataPrototypeMapping.secondToFirstDataTransformation
[constr_1632]	Restriction for firstToSecondDataTransformation and secondToFirstDataTransformation
[constr_1634]	Allowed combinations of ApplicationDataType.category vs. CompuMethod.category
[constr_1635]	Relevance of attribute isOptional
[constr_1636]	Mapping of data types that represent an Optional Element Structure
[constr_1637]	Existence of ImplementationDataTypeElement.isOptional vs. ImplementationDataType.isStructWithOptionalElement
[constr_1638]	First ImplementationDataTypeElement of ImplementationDataType that represents an Optional Element Structure
[constr_1639]	ImplementationDataTypeElement with attribute isOptional set to True
[constr_1640]	No use of Optional Element Structure for interaction with the diagnostic stack
[constr_1662]	Compatibility of ApplicationRecordDataType and ImplementationDataType that both represent an Optional Element Structure

**Table C.54: Added Constraints in R4.4.0**

### C.11.5 Changed Constraints in R4.4.0

Number	Heading
[constr_1006]	applicable data categories
[constr_1007]	Allowed attributes of SwDataDefProps for ApplicationDataTypes



△

Number	Heading
[constr_1009]	SwDataDefProps applicable to ImplementationDataTypes
[constr_1015]	Prioritization of SwDataDefProps
[constr_1024]	Stepwise definition of CompuMethods
[constr_1048]	Compatibility of ApplicationRecordDataTypes
[constr_1050]	Compatibility of ImplementationDataTypes
[constr_1273]	Rules for the initialization of ApplicationArrayType by means of ArrayValueSpecification
[constr_1274]	Rules for the initialization of array-shaped ImplementationDataType by means of ArrayValueSpecification
[constr_1288]	Allowed Attributes vs. category for DataPrototypes typed by ImplementationDataTypes
[constr_1289]	Allowed Attributes vs. category for DataPrototypes typed by ApplicationDataTypes
[constr_1400]	Reference to a specific DataTransformation
[constr_1444]	Limited applicability of Wrapped Union Data Type
[constr_1519]	Existence of attributes vs. category of ApplicationValueSpecification
[constr_2000]	Compatibility of ClientServerOperations triggering the same RunnableEntity

**Table C.55: Changed Constraints in R4.4.0**

### C.11.6 Deleted Constraints in R4.4.0

Number	Heading
[constr_1032]	DelegationSwConnector can only connect PortPrototypes of the same kind
[constr_1297]	Applicability of serverArgumentImplPolicy set to useArrayBaseType
[constr_1443]	category UNION shall not be used for ImplementationDataType
[constr_1515]	Existence of ImplementationDataTypeSubElementRef.implementationDataTypeElement as opposed to ImplementationDataTypeSubElementRef.parameterImplementationDataTypeElement

**Table C.56: Deleted Constraints in R4.4.0**

## C.12 Constraint History of this Document according to AUTOSAR R19-11

### C.12.1 Added Traceables in R19-11

Number	Heading
[TPS_SWCT_01792]	Initialization of a <code>DataPrototype</code> associated with a <code>CompuMethod</code> of category <code>BITFIELD_TEXTTABLE</code>
[TPS_SWCT_01793]	Initialization of a <i>variable-size array</i> typed by an <code>ImplementationDataType</code>
[TPS_SWCT_01794]	Initialization of a <i>variable-size array</i> typed by an <code>ApplicationArrayDataType</code>
[TPS_SWCT_01795]	Further specification to facilitate the association of writing strategies to the corresponding <code>RunnableEntity</code>
[TPS_SWCT_01796]	Prioritization of <code>SwDataDefProps.dataConstr</code> for a <code>DataPrototype</code> of category <code>ARRAY</code>
[TPS_SWCT_01797]	Prioritization of <code>SwDataDefProps.displayFormat</code> for a <code>DataPrototype</code> of category <code>ARRAY</code>
[TPS_SWCT_01798]	Prioritization of <code>SwDataDefProps.stepSize</code> for a <code>DataPrototype</code> of category <code>ARRAY</code>
[TPS_SWCT_01799]	Mapping of bitfields between <code>NvBlockDescriptor</code> and <code>PortPrototype</code>
[TPS_SWCT_01801]	Support for Meta-Data
[TPS_SWCT_01802]	Definition of meta-data in the context of a <code>SenderReceiverInterface</code>
[TPS_SWCT_01803]	<code>MetaDataItems</code> define the same value of attribute <code>length</code>
[TPS_SWCT_01804]	Standardized values of attribute <code>MetaDataItem.metaDataType.value</code>
[TPS_SWCT_01805]	Semantics of the aggregation <code>NvBlockSwComponentType.bulkNvDataDescriptor</code>
[TPS_SWCT_01806]	Simultaneous aggregation of <code>NvBlockSwComponentType.bulkNvDataDescriptor</code> and <code>NvBlockSwComponentType.nvBlockDescriptor</code>
[TPS_SWCT_01807]	Application of <code>NvBlockDataMapping</code> on <code>BulkNvDataDescriptor</code>
[TPS_SWCT_03500]	Status forwarding to data transformer
[TPS_SWCT_03501]	Applicability of status forwarding to data transformer

**Table C.57: Added Traceables in R19-11**



### C.12.2 Changed Traceables in R19-11

Number	Heading
[TPS_SWCT_01029]	<code>AtomicSwComponentType</code> implements a Diagnostic Monitor
[TPS_SWCT_01030]	<code>RunnableEntity</code>
[TPS_SWCT_01132]	<code>AtomicSwComponentType</code> provides information about operating cycles
[TPS_SWCT_01148]	<code>NvBlockDataMapping</code>
[TPS_SWCT_01182]	Conceptual levels for the definition of initial values
[TPS_SWCT_01220]	<code>initValue</code> defines an initial value that shall be taken if the corresponding <code>dataElement</code> has not yet been received
[TPS_SWCT_01243]	Definition of enumeration types
[TPS_SWCT_01309]	signature of a <code>RunnableEntity</code> depends on the connected <code>RTEEvent</code>
[TPS_SWCT_01310]	Categories of <code>RunnableEntitys</code>
[TPS_SWCT_01654]	<code>AtomicSwComponentType</code> offers <code>PortPrototypes</code> typed by <code>Sender-ReceiverInterfaces</code> to adjust the IO signal via diagnostic services
[TPS_SWCT_01656]	Suffix used for the resulting name of the <code>PortInterface</code> for <code>DataServices</code> , <code>IOControlRequest</code> , and <code>IOControlResponse</code>
[TPS_SWCT_01789]	<code>AtomicSwComponentType</code> implements a Diagnostic Monitor that provides monitor data, debouncing by Dem
[TPS_SWCT_01790]	<code>AtomicSwComponentType</code> implements a Diagnostic Monitor that provides monitor data, debouncing by software-component

**Table C.58: Changed Traceables in R19-11**

### C.12.3 Deleted Traceables in R19-11

Number	Heading
[TPS_SWCT_01344]	Consistency of values of <code>timeout</code>
[TPS_SWCT_01429]	[ <code>constr_1135</code> ] only applies for <code>BITFIELD_TEXTTABLE</code>
[TPS_SWCT_01752]	Initialization of a variable-size array

**Table C.59: Deleted Traceables in R19-11**

### C.12.4 Added Constraints in R19-11

Number	Heading
[ <code>constr_1679</code> ]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = signalBasedDiagnostics</code>



△

Number	Heading
[constr_1680]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = AppModeRequestInterface</code>
[constr_1681]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = VerificationStatus</code>
[constr_1682]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xFacVdp</code>
[constr_1683]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xApplRxIndicationCam</code>
[constr_1684]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xApplRxIndicationMapem</code>
[constr_1685]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xApplRxIndicationIvim</code>
[constr_1686]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xApplRxIndicationSpatem</code>
[constr_1694]	Allowed target of <code>SwDataDefProps.implementationDataType</code>
[constr_1706]	Definition of initial value for data transmission
[constr_1712]	Existence of attribute <code>ArrayValueSpecification.intendedPartialInitializationCount</code>
[constr_1713]	<code>NvBlockDescriptor.writingStrategyRole.usedDataElement</code> shall refer to <code>AutosarDataPrototype</code>
[constr_1714]	<code>AutosarDataPrototype</code> shall only be referenced by a single <code>NvBlockDescriptor.writingStrategyRole</code>
[constr_1715]	Possible values of attribute <code>NvBlockDescriptor.writingStrategyRole.role</code>
[constr_1716]	Consistency of attribute <code>NvBlockDescriptor.writingStrategyRole.role</code> set to <code>storeAtShutdown</code>
[constr_1717]	Consistency of attribute <code>NvBlockDescriptor.writingStrategyRole.role</code> set to <code>storeImmediate</code>
[constr_1718]	Inheritance of <code>SwDataDefProps.dataConstr</code> from an array data type to the array elements
[constr_1719]	Inheritance of <code>SwDataDefProps.displayFormat</code> from an array data type to the array elements
[constr_1720]	Inheritance of <code>SwDataDefProps.stepSize</code> from an array data type to the array elements
[constr_1724]	Usage of attribute <code>ClientServerOperation.diagArgIntegrity</code>
[constr_1726]	Ordering of <code>MetaDataItemSet.metaDataItem</code>
[constr_1735]	Limitation of the aggregation of <code>AutosarVariableRef</code> in the context of an <code>NvBlockDataMapping</code> owned by a <code>BulkNvDataDescriptor</code>
[constr_1741]	Restriction to explicit sending semantics for the usage of <code>DataServices</code> in the context of a <code>SwcServiceDependency</code> that aggregates <code>DiagnosticValueNeeds</code> that in turn is referenced by a <code>DiagnosticIoControlNeeds</code>

**Table C.60: Added Constraints in R19-11**

### C.12.5 Changed Constraints in R19-11

Number	Heading
[constr_1051]	Compatibility of <code>SwDataDefProps</code>
[constr_1059]	Compatibility of data types with <code>category VALUE</code>
[constr_1060]	Compatibility of data types with <code>category ARRAY, VAL_BLK</code>
[constr_1061]	Compatibility of data types with <code>category STRUCTURE</code>
[constr_1063]	Compatibility of data types with <code>category BOOLEAN</code>
[constr_1064]	Compatibility of data types with <code>category COM_AXIS, RES_AXIS, CURVE, MAP, CUBOID, CUBE_4, or CUBE_5</code>
[constr_1078]	Compatibility of <code>ClientServerOperations</code>
[constr_1137]	Applicability of <code>ParameterInterface</code>
[constr_1273]	Rules for the initialization of <code>ApplicationArrayDataType</code> by means of <code>ArrayValueSpecification</code>
[constr_1274]	Rules for the initialization of array-shaped <code>ImplementationDataType</code> by means of <code>ArrayValueSpecification</code>
[constr_1607]	Only <code>Wrapped Union Data Types</code> in <code>PortInterface</code>
[constr_1662]	Compatibility of <code>ApplicationRecordDataType</code> and <code>ImplementationDataType</code> that both represent an <code>Optional Element Structure</code>
[constr_2000]	Compatibility of <code>ClientServerOperations</code> triggering the same <code>RunnableEntity</code>
[constr_2013]	Compatibility of <code>ImplementationDataTypes</code> for <code>NvBlockDataMapping</code>

**Table C.61: Changed Constraints in R19-11**

### C.12.6 Deleted Constraints in R19-11

none

## C.13 Constraint History of this Document according to AUTOSAR R20-11

### C.13.1 Added Traceables in R20-11

Number	Heading
[TPS_SWCT_01808]	Dem Service Use Case: software-component checks whether an event is suppressed
[TPS_SWCT_01809]	J1939Dcm wants to retrieve calibration verification numbers from an application software-component





Number	Heading
[TPS_SWCT_01810]	Software-component analyzes predictions about the time synchronization process
[TPS_SWCT_01811]	<code>AtomicSwComponentType</code> reads the current PNC ComM mode
[TPS_SWCT_01812]	Conditional relevance of attribute <code>EndToEndTransformationComSpecProps.disableEndToEndStateMachine</code>
[TPS_SWCT_01813]	Software-Component wants check a certificate on <code>KeyM</code>
[TPS_SWCT_01814]	<code>AtomicSwComponentType</code> wants to retrieve a certificate from <code>KeyM</code>
[TPS_SWCT_01815]	<code>AtomicSwComponentType</code> wants to retrieve elements of a certificate from <code>KeyM</code>
[TPS_SWCT_01816]	<code>AtomicSwComponentType</code> wants to check the existence of a certificate from <code>KeyM</code>
[TPS_SWCT_01817]	<code>AtomicSwComponentType</code> wants to store a (derived) key in <code>KeyM</code>
[TPS_SWCT_01818]	<code>AtomicSwComponentType</code> wants to store a container with encrypted keys (e.g. She-keys) in <code>KeyM</code>
[TPS_SWCT_01819]	<code>AtomicSwComponentType</code> wants to verify if cryptographic operation was executed using a specific key
[TPS_SWCT_01820]	Existence of attribute <code>SenderComSpec.handleOutOfRange</code>
[TPS_SWCT_01821]	Semantics of attribute <code>NotAvailableValueSpecification.defaultPattern</code>
[TPS_SWCT_01822]	Application of attribute <code>NotAvailableValueSpecification.defaultPattern</code> happens only during initialization
[TPS_SWCT_01823]	Definition of <code>ValueSpecification</code> for an <code>ApplicationRecordDataType</code> with unavailable optional elements
[TPS_SWCT_01826]	Application Software Component reports security event
[TPS_SWCT_01827]	Suffix used for the resulting name of the <code>PortInterface</code> for the IdsM Services
[TPS_SWCT_01828]	Application Software Component reports security event using Smart Sensor API
[TPS_SWCT_01829]	Suffix used for the resulting name of the <code>PortInterface</code> for the IdsM Services
[TPS_SWCT_01830]	Application Software Component provides time stamp to IdsM

**Table C.62: Added Traceables in R20-11**

### C.13.2 Changed Traceables in R20-11

Number	Heading
[TPS_SWCT_01004]	Specific default value if <code>serviceKind</code> is not defined
[TPS_SWCT_01078]	Configurable array size
[TPS_SWCT_01178]	Specialized subclasses of <code>ValueSpecification</code>
[TPS_SWCT_01221]	<code>DataFilter</code>





Number	Heading
[TPS_SWCT_01495]	Standardized value of <a href="#">RuleBasedValueSpecification.rule</a>
[TPS_SWCT_01642]	Definition of an “old-world” dynamic-size array data type by means of an <a href="#">ImplementationDataType</a>
[TPS_SWCT_01673]	Application Software Component sends requests using the J1939Rm
[TPS_SWCT_01674]	Application Software Component accepts requests using the J1939Rm
[TPS_SWCT_02003]	<a href="#">AtomicSwComponentType</a> offers <a href="#">PortPrototypes</a> typed by <a href="#">Sender-ReceiverInterfaces</a> or <a href="#">NvDataInterfaces</a> to read/write current values via diagnostic services
[TPS_SWCT_02052]	Definition of Rapid Prototyping Scenario is splittable

**Table C.63: Changed Traceables in R20-11**

### C.13.3 Deleted Traceables in R20-11

Number	Heading
[TPS_SWCT_01039]	Purpose of variant handling
[TPS_SWCT_01510]	The role of pretended networking
[TPS_SWCT_01511]	Configuration option is encoded into <a href="#">ModeDeclaration</a>
[TPS_SWCT_01512]	Request change of <a href="#">Pretended Networking mode</a>
[TPS_SWCT_01513]	React on the change of <a href="#">Pretended Networking mode</a>

**Table C.64: Deleted Traceables in R20-11**

### C.13.4 Added Constraints in R20-11

Number	Heading
[constr_1754]	Aggregation of <a href="#">NumericalRuleBasedValueSpecification</a>
[constr_1755]	Aggregation of <a href="#">CompositeRuleBasedValueSpecification</a>
[constr_1771]	Existence of <a href="#">SwValueCont.unit</a>
[constr_1773]	Value of attribute <a href="#">dataSendPoint.returnValueProvision</a>
[constr_1774]	Value of attribute <a href="#">dataReceivePointByArgument.returnValueProvision</a>
[constr_1775]	Value of attribute <a href="#">serverCallPoint.returnValueProvision</a>
[constr_1776]	Value of attribute <a href="#">asynchronousServerCallResultPoint.returnValueProvision</a>
[constr_1777]	Value of attribute <a href="#">externalTriggeringPoint.returnValueProvision</a>
[constr_1778]	Value of attribute <a href="#">modeSwitchPoint.returnValueProvision</a>
[constr_1779]	Scope of the definition of an <a href="#">AbstractRuleBasedValueSpecification</a>
[constr_1783]	Existence of attribute <a href="#">ImplementationDataTypeElement.arrayImplPolicy</a>





Number	Heading
[constr_1860]	Multiplicity of <a href="#">DelegationSwConnector.innerPort</a>
[constr_1861]	Multiplicity of <a href="#">DelegationSwConnector.outerPort</a>
[constr_1862]	Multiplicity of <a href="#">PassThroughSwConnector.requiredOuterPort</a>
[constr_1863]	Multiplicity of <a href="#">PassThroughSwConnector.providedOuterPort</a>
[constr_1864]	Multiplicity of <a href="#">InstantiationRTEEventProps.refinedEvent</a>
[constr_1865]	Existence of <a href="#">InvalidationPolicy.dataElement</a>
[constr_1866]	Existence of <a href="#">MetaDataItem.length</a>
[constr_1867]	Existence of <a href="#">MetaDataItem.metaDataType</a>
[constr_1868]	Existence of <a href="#">MetaDataItemSet.dataElement</a>
[constr_1869]	Existence of attribute <a href="#">ArgumentDataPrototype.direction</a>
[constr_1870]	Existence of attribute <a href="#">ApplicationError.errorCode</a>
[constr_1871]	Existence of attribute <a href="#">ModeRequestTypeMap.implementationDataType</a>
[constr_1872]	Existence of attribute <a href="#">ModeRequestTypeMap.modeGroup</a>
[constr_1873]	Existence of <a href="#">DataPrototypeMapping.firstDataPrototype</a>
[constr_1874]	Existence of <a href="#">DataPrototypeMapping.secondDataPrototype</a>
[constr_1875]	Existence of reference <a href="#">ClientServerOperationMapping.firstOperation</a>
[constr_1876]	Existence of reference <a href="#">ClientServerOperationMapping.secondOperation</a>
[constr_1877]	Existence of reference <a href="#">ModeDeclarationGroupPrototypeMapping.first-ModeGroup</a>
[constr_1878]	Existence of reference <a href="#">ModeDeclarationGroupPrototypeMapping.second-ModeGroup</a>
[constr_1879]	Existence of reference <a href="#">ModeDeclarationMapping.firstMode</a>
[constr_1880]	Existence of reference <a href="#">ModeDeclarationMapping.secondMode</a>
[constr_1881]	Existence of reference <a href="#">TriggerMapping.firstTrigger</a>
[constr_1882]	Existence of reference <a href="#">TriggerMapping.secondTrigger</a>
[constr_1883]	Existence of <a href="#">ApplicationCompositeDataTypeSubElementRef.applicationCompositeElement</a>
[constr_1884]	Existence of attribute <a href="#">TextTableMapping.identicalMapping</a>
[constr_1885]	Existence of attribute <a href="#">TextTableMapping.mappingDirection</a>
[constr_1886]	Existence of attribute <a href="#">TextTableValuePair.firstValue</a>
[constr_1887]	Existence of attribute <a href="#">TextTableValuePair.secondValue</a>
[constr_1888]	Existence of attribute <a href="#">DataTransformation.executeDespiteDataUnavailability</a>
[constr_1889]	Existence of attribute <a href="#">QueuedReceiverComSpec.queueLength</a>
[constr_1890]	Existence of attribute <a href="#">DataFilter.dataFilterType</a>
[constr_1891]	Existence of attribute <a href="#">NonqueuedReceiverComSpec.initValue</a>
[constr_1892]	Existence of attribute <a href="#">TransmissionAcknowledgementRequest.timeout</a>
[constr_1893]	Existence of attribute <a href="#">ServerComSpec.queueLength</a>
[constr_1894]	Existence of attribute <a href="#">ModeSwitchSenderComSpec.queueLength</a>
[constr_1895]	Existence of attribute <a href="#">ModeSwitchSenderComSpec.modeGroup</a>





Number	Heading
[constr_1896]	Existence of attribute <a href="#">ModeSwitchReceiverComSpec.modeGroup</a>
[constr_1897]	Existence of reference <a href="#">ParameterProvideComSpec.parameter</a>
[constr_1898]	Existence of reference <a href="#">ParameterRequireComSpec.parameter</a>
[constr_1899]	Existence of reference <a href="#">NvRequireComSpec.variable</a>
[constr_1900]	Existence of reference <a href="#">NvProvideComSpec.variable</a>
[constr_1901]	Existence of attribute <a href="#">EndToEndDescription.category</a>
[constr_1902]	Existence of attribute <a href="#">EndToEndProtection.endToEndProfile</a>
[constr_1903]	Existence of reference <a href="#">DataTypeMap.applicationDataType</a>
[constr_1904]	Existence of reference <a href="#">DataTypeMap.implementationDataType</a>
[constr_1905]	Existence of attribute <a href="#">SwTextProps.arraySizeSemantics</a>
[constr_1906]	Existence of attribute <a href="#">SwTextProps.swMaxTextSize</a>
[constr_1907]	Existence of attribute <a href="#">ApplicationArrayDataType.element</a>
[constr_1908]	Existence of attribute <a href="#">ApplicationRecordDataType.element</a>
[constr_1909]	Existence of attribute <a href="#">ImplementationProps.symbol</a>
[constr_1910]	Existence of attribute <a href="#">BaseType.baseTypeDefinition</a>
[constr_1911]	Existence of <a href="#">ArVariableInImplementationDataInstanceRef.targetDataPrototype</a>
[constr_1912]	Existence of reference <a href="#">ArParameterInImplementationDataInstanceRef.targetDataPrototype</a>
[constr_1913]	Existence of attribute <a href="#">CompuRationalCoeffs.compuDenominator</a>
[constr_1914]	Existence of attribute <a href="#">CompuRationalCoeffs.compuNumerator</a>
[constr_1915]	Existence of attribute <a href="#">PhysicalDimensionMapping.firstPhysicalDimension</a>
[constr_1916]	Existence of attribute <a href="#">PhysicalDimensionMapping.secondPhysicalDimension</a>
[constr_1917]	Existence of <a href="#">ConstantSpecification.valueSpec</a>
[constr_1918]	Existence of <a href="#">RecordValueSpecification.field</a>
[constr_1919]	Existence of <a href="#">TextValueSpecification.value</a>
[constr_1920]	Existence of <a href="#">NumericalValueSpecification.value</a>
[constr_1921]	Existence of <a href="#">ReferenceValueSpecification.referenceValue</a>
[constr_1922]	Existence of <a href="#">ApplicationRuleBasedValueSpecification.category</a>
[constr_1923]	Existence of <a href="#">RuleBasedAxisCont.ruleBasedValues</a>
[constr_1924]	Existence of <a href="#">RuleBasedValueCont.ruleBasedValues</a>
[constr_1925]	Existence of <a href="#">NumericalRuleBasedValueSpecification.ruleBasedValues</a>
[constr_1926]	Existence of <a href="#">RuleBasedValueSpecification.rule</a>
[constr_1927]	Existence of <a href="#">RuleBasedValueSpecification.arguments</a>
[constr_1928]	Existence of <a href="#">CompositeRuleBasedValueSpecification.rule</a>
[constr_1929]	Existence of <a href="#">CompositeRuleBasedValueSpecification.argument</a>
[constr_1930]	Existence of <a href="#">ConstantReference.constant</a>
[constr_1931]	Existence of <a href="#">ConstantSpecificationMapping.applConstant</a>





Number	Heading
[constr_1932]	Existence of <a href="#">ConstantSpecificationMapping.implConstant</a>
[constr_1933]	Existence of <a href="#">CalibrationParameterValue.initializedParameter</a>
[constr_1934]	Existence of attribute <a href="#">SwcInternalBehavior.handleTerminationAndRestart</a>
[constr_1935]	Existence of attribute <a href="#">SwcInternalBehavior.supportsMultipleInstantiation</a>
[constr_1936]	Existence of attribute <a href="#">RunnableEntity.symbol</a>
[constr_1937]	Existence of attribute <a href="#">TimingEvent.period</a>
[constr_1938]	Existence of attribute <a href="#">RunnableEntityArgument.symbol</a>
[constr_1939]	Existence of attribute <a href="#">ExecutableEntityActivationReason.bitPosition</a>
[constr_1940]	Existence of attribute <a href="#">AsynchronousServerCallReturnsEvent.eventSource</a>
[constr_1941]	Existence of attribute <a href="#">DataSendCompletedEvent.eventSource</a>
[constr_1942]	Existence of attribute <a href="#">DataWriteCompletedEvent.eventSource</a>
[constr_1943]	Existence of attribute <a href="#">DataReceivedEvent.data</a>
[constr_1944]	Existence of attribute <a href="#">DataReceiveErrorEvent.data</a>
[constr_1945]	Existence of attribute <a href="#">OperationInvokedEvent.operation</a>
[constr_1946]	Existence of attribute <a href="#">SwcModeSwitchEvent.activation</a>
[constr_1947]	Existence of reference <a href="#">SwcModeSwitchEvent.mode</a>
[constr_1948]	Existence of attribute <a href="#">ModeSwitchedAckEvent.eventSource</a>
[constr_1949]	Existence of attribute <a href="#">ExternalTriggerOccurredEvent.trigger</a>
[constr_1950]	Existence of attribute <a href="#">InternalTriggerOccurredEvent.eventSource</a>
[constr_1951]	Existence of attribute <a href="#">WaitPoint.timeout</a>
[constr_1952]	Existence of reference <a href="#">WaitPoint.trigger</a>
[constr_1953]	Existence of attribute <a href="#">SwcExclusiveAreaPolicy.apiPrinciple</a>
[constr_1954]	Existence of attribute <a href="#">VariableAccess.accessedVariable</a>
[constr_1955]	Existence of attribute <a href="#">ServerCallPoint.operation</a>
[constr_1956]	Existence of attribute <a href="#">ServerCallPoint.timeout</a>
[constr_1957]	Existence of attribute <a href="#">AsynchronousServerCallResultPoint.asynchronousServerCallPoint</a>
[constr_1958]	Existence of attribute <a href="#">ParameterAccess.accessedParameter</a>
[constr_1959]	Existence of attribute <a href="#">InstantiationDataDefProps.swDataDefProps</a>
[constr_1960]	Existence of attribute <a href="#">PortAPIOption.port</a>
[constr_1961]	Existence of attribute <a href="#">PortDefinedArgumentValue.value</a>
[constr_1962]	Existence of attribute <a href="#">PortDefinedArgumentValue.valueType</a>
[constr_1963]	Existence of attribute <a href="#">CommunicationBufferLocking.supportBufferLocking</a>
[constr_1964]	Existence of attribute <a href="#">PerInstanceMemory.type</a>
[constr_1965]	Existence of attribute <a href="#">PerInstanceMemory.typeDefinition</a>
[constr_1966]	Existence of attribute <a href="#">Implementation.swVersion</a>
[constr_1967]	Existence of attribute <a href="#">Implementation.vendorId</a>
[constr_1968]	Existence of attribute <a href="#">Implementation.codeDescriptor</a>







Number	Heading
[constr_1969]	Existence of attribute <a href="#">SwcImplementation.behavior</a>
[constr_1970]	Existence of attribute <a href="#">PerInstanceMemorySize.alignment</a>
[constr_1971]	Existence of attribute <a href="#">PerInstanceMemorySize.perInstanceMemory</a>
[constr_1972]	Existence of attribute <a href="#">PerInstanceMemorySize.size</a>
[constr_1973]	Existence of attribute <a href="#">ModeDeclarationGroup.initialMode</a>
[constr_1974]	Existence of attribute <a href="#">ModeDeclarationGroup.modeDeclaration</a>
[constr_1975]	Existence of attribute <a href="#">ModeTransition.enteredMode</a>
[constr_1976]	Existence of attribute <a href="#">ModeTransition.exitedMode</a>
[constr_1977]	Existence of attribute <a href="#">ModeErrorBehavior.errorReactionPolicy</a>
[constr_1978]	Existence of attribute <a href="#">SwcModeManagerErrorEvent.modeGroup</a>
[constr_1979]	Existence of the reference <a href="#">SwcBswMapping.bswBehavior</a>
[constr_1980]	Existence of the reference <a href="#">SwcBswMapping.swcBehavior</a>
[constr_1981]	Existence of attribute <a href="#">NvBlockDescriptor.nvBlockNeeds</a>
[constr_1982]	Existence of attribute <a href="#">ModeSwitchEventTriggeredActivity.role</a>
[constr_1983]	Existence of attribute <a href="#">ModeSwitchEventTriggeredActivity.swcModeSwitchEvent</a>
[constr_1984]	Existence of instance reference <a href="#">NvBlockDataMapping.nvRamBlockElement</a>
[constr_1985]	Existence of the reference <a href="#">SupervisedEntityNeeds.toleratedFailedCycles</a>
[constr_1986]	Existence of the reference <a href="#">DiagnosticRoutineNeeds.diagRoutineType</a>
[constr_1987]	Existence of instance reference <a href="#">RapidPrototypingScenario.hostSystem</a>
[constr_1988]	Existence of attribute <a href="#">RptProfile.maxServicePointId</a>
[constr_1989]	Existence of attribute <a href="#">RptProfile.minServicePointId</a>
[constr_1990]	Existence of attribute <a href="#">RptProfile.servicePointSymbolPost</a>
[constr_1991]	Existence of attribute <a href="#">RptProfile.servicePointSymbolPre</a>
[constr_1992]	Existence of attribute <a href="#">RptProfile.stimEnabler</a>
[constr_1993]	Existence of attribute <a href="#">RptImplPolicy.rptEnablerImplType</a>
[constr_1994]	Existence of attribute <a href="#">RptImplPolicy.rptPreparationLevel</a>
[constr_1995]	Existence of attribute <a href="#">RptSwPrototypingAccess.rptHookAccess</a>
[constr_1996]	Existence of attribute <a href="#">RptSwPrototypingAccess.rptReadAccess</a>
[constr_1997]	Existence of attribute <a href="#">RptSwPrototypingAccess.rptWriteAccess</a>
[constr_1998]	Existence of attribute <a href="#">RptExecutableEntityProperties.maxRptEventId</a>
[constr_1999]	Existence of attribute <a href="#">RptExecutableEntityProperties.minRptEventId</a>
[constr_5234]	Existence of attribute <a href="#">E2EProfileCompatibilityProps.transitToInvalidExtended</a> is mandatory for each <a href="#">EndToEndTransformationComSpecProps</a>
[constr_10000]	Existence of attribute <a href="#">RptExecutableEntityProperties.rptExecutionControl</a>
[constr_10001]	Existence of attribute <a href="#">RptExecutableEntityProperties.rptServicePoint</a>
[constr_10005]	Existence of attribute <a href="#">NotAvailableValueSpecification.defaultPattern</a>





Number	Heading
[constr_10006]	Valid interval of attribute <code>NotAvailableValueSpecification.defaultPattern</code>
[constr_10009]	Aggregation of <code>ApplicationRuleBasedValueSpecification</code>
[constr_10016]	Applicability of <code>OsTaskExecutionEvent</code>
[constr_10017]	Existence of attribute <code>SwAxisCont.category</code>
[constr_10018]	Existence of attribute <code>SwAxisCont.swAxisIndex</code>
[constr_10019]	Existence of attribute <code>SwAxisCont.swValuesPhys</code>
[constr_10020]	Existence of attribute <code>RoleBasedDataTypeAssignment.usedImplementationDataType</code>
[constr_10028]	Existence of reference stereotyped <code>&lt;&lt;isOfType&gt;&gt;</code>

**Table C.65: Added Constraints in R20-11**

### C.13.5 Changed Constraints in R20-11

Number	Heading
[constr_1051]	Compatibility of <code>SwDataDefProps</code>
[constr_1060]	Compatibility of data types with <code>category ARRAY, VAL_BLK</code>
[constr_1066]	Forbidden mappings to <code>ImplementationDataType</code>
[constr_1197]	Existence of <code>compositeNetworkRepresentation</code> shall be comprehensive
[constr_1224]	<code>DataPrototype</code> is typed by an <code>ApplicationArrayDataType</code>
[constr_1289]	Allowed Attributes vs. <code>category</code> for <code>DataPrototypes</code> typed by <code>ApplicationDataTypes</code>
[constr_1296]	<code>DataPrototypes</code> used as <code>explicitInterRunnableVariable</code> or <code>implicitInterRunnableVariable</code> and <code>category DATA_REFERENCE</code>
[constr_1393]	Existence of <code>RuleBasedValueCont.unit</code>
[constr_1397]	Existence of attributes of <code>TransformerHardErrorEvent</code>
[constr_1422]	Value of <code>category</code> is <code>VOID</code>
[constr_1741]	Restriction to explicit sending semantics for the usage of <code>DataServices</code> in the context of a <code>SwcServiceDependency</code> that aggregates <code>DiagnosticValueNeeds</code> that in turn is referenced by a <code>DiagnosticIoControlNeeds</code>
[constr_2000]	Compatibility of <code>ClientServerOperations</code> triggering the same <code>RunnableEntity</code>
[constr_2034]	<code>SwAddrMethod</code> referenced by <code>RunnableEntitys</code> , <code>BswCalledEntitys</code> , or <code>BswSchedulableEntitys</code>
[constr_2052]	Values of <code>swArraysize</code> and the number of values provided by <code>swValuesPhys</code> shall be consistent.
[constr_2058]	Mandatory information of a <code>RuleBasedValueCont</code>

**Table C.66: Changed Constraints in R20-11**

### C.13.6 Deleted Constraints in R20-11

none

## C.14 Constraint History of this Document according to AUTOSAR R21-11

### C.14.1 Added Traceables in R21-11

Number	Heading
[TPS_SWCT_01831]	Dcm can directly access <a href="#">SenderReceiverInterface.dataElements</a> or <a href="#">NvDataInterface.nvDatas</a> in <a href="#">AbstractRequiredPortPrototypes</a>
[TPS_SWCT_01832]	<a href="#">SecOc</a> Use Case: Receive notification about an authentication attempt
[TPS_SWCT_01833]	Semantics of <a href="#">ServiceDependency.diagnosticRelevance</a>
[TPS_SWCT_01834]	<a href="#">invalidValue</a> is inside the scope of the <a href="#">compuMethod</a>
[TPS_SWCT_01835]	<a href="#">invalidValue</a> is outside the scope of the <a href="#">compuMethod</a>
[TPS_SWCT_01836]	Attributes of <a href="#">CompositeRuleBasedValueSpecification</a>
[TPS_SWCT_01837]	Types for record layouts
[TPS_SWCT_01838]	<a href="#">ValueSpecification</a> shall fit into data type
[TPS_SWCT_01839]	Size of Compound Primitive Data Type is variant
[TPS_SWCT_01840]	A <a href="#">ParameterSwComponentType</a> references a <a href="#">ConstantSpecificationMappingSet</a>
[TPS_SWCT_01841]	A <a href="#">NvBlockSwComponentType</a> references a <a href="#">ConstantSpecificationMappingSet</a>
[TPS_SWCT_01842]	Applicability of constraints of <a href="#">CompuScales</a>
[TPS_SWCT_01843]	Value of <a href="#">PassThroughSwConnector.category</a>
[TPS_SWCT_01844]	Optional method arguments

Table C.67: Added Traceables in R21-11

### C.14.2 Changed Traceables in R21-11

Number	Heading
[TPS_SWCT_01049]	Two ways to use the <a href="#">ExclusiveAreas</a>
[TPS_SWCT_01195]	Mapping of composite element to primitive <a href="#">DataPrototype</a>
[TPS_SWCT_01253]	Rules apply for the usage of the attribute <a href="#">ImplementationDataType.typeEmitter</a>
[TPS_SWCT_01314]	<a href="#">RTEEvent</a>
[TPS_SWCT_01551]	Mapping of elements on the “source” end to elements on the “target” end





Number	Heading
[TPS_SWCT_01579]	Dcm can directly access <code>SenderReceiverInterface.dataElements</code> , <code>NvDataInterface.nvDataS</code> , or <code>ParameterInterface.parameters</code> in <code>AbstractProvidedPortPrototype</code>
[TPS_SWCT_01580]	Dem can directly access <code>SenderReceiverInterface.dataElements</code> , <code>NvDataInterface.nvDataS</code> , or <code>ParameterInterface.parameters</code> in <code>PPortPrototypes</code>
[TPS_SWCT_01586]	Writing strategies for <i>nv data</i>
[TPS_SWCT_01795]	Further specification to facilitate the association of writing strategies to the corresponding <code>RunnableEntity</code>
[TPS_SWCT_02011]	<code>AtomicSwComponentType</code> offers a client port to read DTR value

**Table C.68: Changed Traceables in R21-11**

### C.14.3 Deleted Traceables in R21-11

Number	Heading
[TPS_SWCT_01591]	Existence of attribute <code>DiagnosticEventNeeds.reportBehavior</code>
[TPS_SWCT_01697]	Supported development approach for software-components that interact with AUTOSAR services
[TPS_SWCT_01698]	Attributes that are subject to development approach

**Table C.69: Deleted Traceables in R21-11**

### C.14.4 Added Constraints in R21-11

Number	Heading
[constr_10032]	Restrictions for the usage of <code>ServiceDependency.diagnosticRelevance</code>
[constr_10033]	Existence of <code>MemorySection.swAddrmethod</code>
[constr_10034]	Existence of <code>MemorySection.alignment</code>
[constr_10040]	Value of <code>ApplicationValueSpecification.swAxisCont.category</code>
[constr_10041]	Value of <code>ApplicationRuleBasedValueSpecification.swAxisCont.category</code>
[constr_10067]	Creation of <code>AssemblySwConnector</code> for service communication
[constr_10068]	Standardized values for <code>SectionInitializationPolicyType</code>
[constr_10071]	Allowed multiplicities of <code>SenderComSpec</code> attributes for communication between <code>ApplicationSwComponentType</code> and <code>NvBlockSwComponentType</code>
[constr_10072]	Allowed multiplicities of <code>SenderComSpec</code> attributes for communication between <code>NvBlockSwComponentType</code> and <code>ApplicationSwComponentType</code>
[constr_10073]	Existence of <code>DataReceiveErrorEvent</code>





Number	Heading
[constr_10074]	Consistency of attribute <code>NvBlockDescriptor.writingStrategy.role</code> set to <code>storeOnChange</code>
[constr_10075]	Existence of <code>CompositeRuleBasedValueSpecification.argument</code> vs. <code>compoundPrimitiveArgument</code>
[constr_10087]	Restriction for the existence of a <code>SubElementMapping</code>

**Table C.70: Added Constraints in R21-11**

### C.14.5 Changed Constraints in R21-11

Number	Heading
[constr_1004]	Mapping of <code>ApplicationDataTypes</code> in the scope of single <code>AtomicSwComponentTypes</code>
[constr_1005]	Compatibility of <code>ImplementationDataTypes</code> mapped to the same <code>ApplicationDataType</code>
[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_1010]	If <code>nativeDeclaration</code> does not exist
[constr_1011]	<code>category</code> of <code>SwBaseType</code>
[constr_1012]	Value of <code>category</code> is <code>FIXED_LENGTH</code>
[constr_1014]	Supported value encodings for <code>SwBaseType</code>
[constr_1015]	Prioritization of <code>SwDataDefProps</code>
[constr_1016]	Restriction of <code>invalidValue</code> for <code>ImplementationDataType</code> and <code>ImplementationDataTypeElement</code>
[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_1020]	<code>ParameterDataPrototype</code> needs to be of compatible data type as referenced in <code>sharedAxisType</code>
[constr_1022]	Limits shall be defined for each direction of <code>CompuMethod</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_1029]	<code>ConstantSpecificationMapping</code> and <code>ConstantSpecification</code>
[constr_1033]	Communication scenarios for sender/receiver communication
[constr_1035]	Recursive definition of <code>CompositionSwComponentType</code>
[constr_1036]	Connect kinds of <code>PortInterfaces</code>
[constr_1037]	Client shall not be connected to multiple servers
[constr_1038]	Reference to <code>ApplicationError</code>





Number	Heading
[constr_1039]	Relevance of <code>swImplPolicy</code>
[constr_1040]	Conversion of <code>SenderReceiverInterfaces</code>
[constr_1041]	Conversion of <code>ClientServerInterfaces</code>
[constr_1043]	<code>PortInterface</code> vs. <code>ComSpec</code>
[constr_1044]	Applicability of <code>DataFilter</code>
[constr_1045]	Supported value encodings for <code>SwBaseType</code> in the context of <code>PortInterfaces</code>
[constr_1046]	Applicability of [constr_1045]
[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_1051]	Compatibility of <code>SwDataDefProps</code>
[constr_1052]	Compatibility of <code>Units</code>
[constr_1053]	Compatibility of <code>PhysicalDimensions</code>
[constr_1054]	No <code>DataConstr</code> available at the provider
[constr_1055]	<code>ImplementationDataType</code> has category <code>VALUE</code>
[constr_1056]	<code>ImplementationDataType</code> has category <code>TYPE_REFERENCE</code>
[constr_1057]	<code>ImplementationDataType</code> has category <code>DATA_REFERENCE</code>
[constr_1058]	<code>ImplementationDataType</code> has category <code>FUNCTION_REFERENCE</code>
[constr_1059]	Compatibility of data types with category <code>VALUE</code>
[constr_1060]	Compatibility of data types with category <code>ARRAY</code> , <code>VAL_BLK</code>
[constr_1061]	Compatibility of data types with category <code>STRUCTURE</code>
[constr_1063]	Compatibility of data types with category <code>BOOLEAN</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_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_1071]	compatibility of <code>ParameterDataPrototype</code> and <code>VariableDataPrototype</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_1076]	Compatibility of <code>ArgumentDataPrototypes</code>





Number	Heading
[constr_1077]	Compatibility of <a href="#">ApplicationErrors</a>
[constr_1078]	Compatibility of <a href="#">ClientServerOperations</a>
[constr_1079]	Compatibility of <a href="#">ClientServerInterfaces</a> in the context of an <a href="#">AssemblySwConnector</a>
[constr_1080]	Compatibility of <a href="#">ClientServerInterfaces</a> in the context of an <a href="#">DelegationSwConnector</a>
[constr_1081]	Compatibility of <a href="#">TriggerInterfaces</a> in the context of an <a href="#">AssemblySwConnector</a>
[constr_1082]	Compatibility of <a href="#">TriggerInterfaces</a> in the context of an <a href="#">DelegationSwConnector</a>
[constr_1083]	Compatibility of <a href="#">Triggers</a>
[constr_1084]	delegation of a provided outer <a href="#">PortPrototype</a>
[constr_1086]	<a href="#">SwConnector</a> between two specific <a href="#">PortPrototypes</a>
[constr_1087]	<a href="#">AssemblySwConnector</a> inside <a href="#">CompositionSwComponentType</a>
[constr_1088]	<a href="#">DelegationSwConnector</a> inside <a href="#">CompositionSwComponentType</a>
[constr_1092]	<a href="#">ParameterSwComponentType</a>
[constr_1093]	Definition of textual strings
[constr_1095]	Values of <a href="#">nDataSets</a> vs. <a href="#">reliability</a>
[constr_1096]	<a href="#">SwcModeSwitchEvent</a> and <a href="#">WaitPoint</a>
[constr_1097]	<a href="#">RunnableEntity</a> that has a <a href="#">WaitPoint</a>
[constr_1098]	Mode switch and mode disabling
[constr_1100]	Unconnected <a href="#">RPortPrototype</a> typed by a <a href="#">DataInterface</a>
[constr_1101]	Mode-related communication
[constr_1102]	<a href="#">ApplicationError</a> in the scope of one <a href="#">SwComponentType</a>
[constr_1103]	<a href="#">NonqueuedReceiverComSpec</a> and <a href="#">enableUpdate</a>
[constr_1104]	Trigger sink and trigger source
[constr_1105]	Value of <a href="#">arraySize</a>
[constr_1106]	Structure shall have at least one element
[constr_1107]	Union shall have at least one element
[constr_1108]	Value of <a href="#">ApplicationError.errorCode</a>
[constr_1109]	Mapping of <a href="#">SwComponentPrototypes</a> typed by a <a href="#">SensorActuatorSwComponentType</a>
[constr_1111]	Constraints of <a href="#">dataId</a> in <a href="#">PROFILE_01</a>
[constr_1112]	Constraints of <a href="#">dataIdMode</a> in <a href="#">PROFILE_01</a>
[constr_1113]	Existence of attributes in <a href="#">PROFILE_01</a>
[constr_1114]	Constraints of <a href="#">crcOffset</a> in <a href="#">PROFILE_01</a>
[constr_1115]	Constraints of <a href="#">counterOffset</a> in <a href="#">PROFILE_01</a>
[constr_1116]	Constraints of <a href="#">dataLength</a> in <a href="#">PROFILE_01</a>
[constr_1117]	Constraints of <a href="#">maxDeltaCounterInit</a> in <a href="#">PROFILE_01</a>
[constr_1118]	Existence of attributes in <a href="#">PROFILE_02</a>





Number	Heading
[constr_1119]	Constraints of <code>dataLength</code> in <code>PROFILE_02</code>
[constr_1120]	Constraints of <code>dataId</code> in <code>PROFILE_02</code>
[constr_1121]	Constraints of <code>maxDeltaCounterInit</code> in <code>PROFILE_02</code>
[constr_1126]	Compatibility of <code>DataConstrs</code>
[constr_1128]	Queue length of <code>ClientServerOperations</code> associated with the same <code>RunnableEntity</code>
[constr_1129]	<code>swImplPolicy</code> and <code>NonqueuedReceiverComSpec</code>
[constr_1130]	<code>swImplPolicy</code> and <code>QueuedReceiverComSpec</code>
[constr_1131]	<code>swImplPolicy</code> and <code>NonqueuedSenderComSpec</code>
[constr_1132]	<code>swImplPolicy</code> and <code>QueuedSenderComSpec</code>
[constr_1134]	Allowed structure of <code>TEXTTABLE</code>
[constr_1135]	Limit of <code>vt</code> in <code>BITFIELD_TEXTTABLE</code>
[constr_1137]	Applicability of <code>ParameterInterface</code>
[constr_1138]	<code>assignedPort</code> and <code>DiagEventDebounceMonitorInternal</code>
[constr_1139]	<code>assignedPort</code> of <code>DiagEventDebounceMonitorInternal</code> shall refer to an <code>RPortPrototype</code>
[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_1144]	<code>SensorActuatorSwComponentType</code> , <code>EcuAbstractionSwComponentType</code> , and <code>ComplexDeviceDriverSwComponentType</code> may only reference a <code>HwType</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_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_1158]	Applicable <code>categorys</code> for attribute <code>ImplementationDataType</code> . <code>swDataDefProps.compuMethod</code>
[constr_1159]	Consistency of <code>VariableAndParameterInterfaceMapping</code> with respect to the referenced <code>DataInterfaces</code>
[constr_1161]	Applicability of the attribute <code>Ref.index</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>







Number	Heading
[constr_1165]	Applicability of <a href="#">RunnableEntityArgument</a>
[constr_1166]	Restrictions of <a href="#">ModeRequestTypeMap</a>
[constr_1167]	<a href="#">ImplementationDataTypes</a> used as <a href="#">ModeRequestTypeMap</a> . <a href="#">implementationDataType</a>
[constr_1168]	Compatibility of <a href="#">ImplementationDataTypes</a> used in the <a href="#">ModeRequestTypeMap</a>
[constr_1169]	Allowed values for <a href="#">Trigger.swImplPolicy</a>
[constr_1172]	Allowed values of <a href="#">SwCalibrationAccessEnum</a> for <a href="#">ModeDeclarationGroupPrototype</a>
[constr_1173]	Applicability of <a href="#">AutosarParameterRef</a> referencing a <a href="#">VariableDataPrototype</a>
[constr_1174]	<a href="#">PortInterfaces</a> used in the context of <a href="#">CompositionSwComponentTypes</a> cannot refer to AUTOSAR services
[constr_1175]	Depending on its <a href="#">category</a> , <a href="#">CompuMethod</a> shall refer to a <a href="#">unit</a>
[constr_1176]	Compatibility of <a href="#">CompuScales</a> of <a href="#">category</a> <a href="#">LINEAR</a> and <a href="#">RAT_FUNC</a>
[constr_1177]	Allowed <a href="#">targetCategory</a> for <a href="#">SwPointerTargetProps</a>
[constr_1178]	Existence of attributes of <a href="#">SwDataDefProps</a> in the context of <a href="#">ImplementationDataType</a>
[constr_1181]	Numerical values used in <a href="#">ModeDeclaration.value</a> and <a href="#">ModeDeclarationGroup.onTransitionValue</a>
[constr_1182]	Allowed values for <a href="#">InternalTriggeringPoint.swImplPolicy</a>
[constr_1183]	<a href="#">EndToEndProtectionVariablePrototypes</a> aggregated by <a href="#">EndToEndProtection</a>
[constr_1184]	Consistency of <a href="#">rootDataPrototype</a> and <a href="#">base</a> in the context of <a href="#">ApplicationCompositeElementInPortInterfaceInstanceRef</a>
[constr_1185]	Consistency of data types in the context of <a href="#">ApplicationCompositeElementInPortInterfaceInstanceRef</a>
[constr_1186]	Consistency of data types in the context of <a href="#">ArVariableInImplementationDataInstanceRef</a>
[constr_1187]	Compatibility of <a href="#">VariableDataPrototypes</a> or <a href="#">ParameterDataPrototypes</a> typed by composite data types
[constr_1188]	Existence of <a href="#">ReceiverComSpec.replaceWith</a>
[constr_1191]	Value of <a href="#">Limit</a> shall yield a numerical value
[constr_1192]	Compatibility of “ <a href="#">IDENTICAL</a> ” to “ <a href="#">RAT_FUNC</a> ” or “ <a href="#">LINEAR</a> ”
[constr_1193]	<a href="#">ModeDeclaration</a> shall be referenced by at least one <a href="#">ModeTransition</a> in the role <a href="#">enteredMode</a>
[constr_1194]	Identical <a href="#">ModeTransitions</a>
[constr_1195]	<a href="#">SwcModeSwitchEvent</a> and the definition of <a href="#">ModeTransition</a>
[constr_1196]	Existence of <a href="#">networkRepresentation</a> vs. <a href="#">compositeNetworkRepresentation</a>
[constr_1197]	Existence of <a href="#">compositeNetworkRepresentation</a> shall be comprehensive
[constr_1200]	Queued communication is not applicable for <a href="#">dataElements</a> owned by <a href="#">PRPortPrototype</a>
[constr_1202]	Supported connections by <a href="#">AssemblySwConnector</a> for <a href="#">PortPrototypes</a> typed by a <a href="#">SenderReceiverInterface</a> or <a href="#">NvDataInterface</a>





Number	Heading
[constr_1203]	Supported connections by <a href="#">DelegationSwConnector</a> for <a href="#">PortPrototypes</a> typed by a <a href="#">SenderReceiverInterface</a> or <a href="#">NvDataInterface</a>
[constr_1204]	Supported connections by <a href="#">AssemblySwConnector</a> for <a href="#">PortPrototypes</a> typed by a <a href="#">ClientServerInterface</a> , <a href="#">ModeSwitchInterface</a> , or <a href="#">TriggerInterface</a>
[constr_1205]	Supported connections by <a href="#">DelegationSwConnector</a> for <a href="#">PortPrototypes</a> typed by a <a href="#">ClientServerInterface</a> , <a href="#">ModeSwitchInterface</a> , or <a href="#">TriggerInterface</a>
[constr_1209]	Mapping of <a href="#">ModeDeclarations</a> of mode user to <a href="#">ModeDeclaration</a> of mode manager
[constr_1210]	Mapping of <a href="#">ModeDeclarations</a> of mode user to <b>all</b> <a href="#">ModeDeclarations</a> of mode manager
[constr_1211]	Constraints of <a href="#">maxNoNewOrRepeatedData</a> in <a href="#">PROFILE_01</a>
[constr_1212]	Constraints of <a href="#">syncCounterInit</a> in <a href="#">PROFILE_01</a>
[constr_1213]	Constraints of <a href="#">maxNoNewOrRepeatedData</a> in <a href="#">PROFILE_02</a>
[constr_1214]	Constraints of <a href="#">syncCounterInit</a> in <a href="#">PROFILE_02</a>
[constr_1219]	Invalidation depends on the value of <a href="#">swImplPolicy</a>
[constr_1220]	Compatibility of <a href="#">SwBaseType</a>
[constr_1221]	<a href="#">DataPrototype</a> is typed by an <a href="#">ApplicationPrimitiveDataType</a>
[constr_1222]	<a href="#">category</a> of an <a href="#">AutosarDataType</a> used to type a <a href="#">DataPrototype</a> is set to <a href="#">STRING</a>
[constr_1223]	<a href="#">DataPrototype</a> is typed by an <a href="#">ApplicationRecordDataType</a>
[constr_1224]	<a href="#">DataPrototype</a> is typed by an <a href="#">ApplicationArrayDataType</a>
[constr_1225]	<a href="#">DataPrototype</a> is typed by an <a href="#">ImplementationDataType</a> that references a <a href="#">CompuMethod</a> of category <a href="#">TEXTTABLE</a> or <a href="#">BITFIELD_TEXTTABLE</a>
[constr_1226]	Applicable range for <a href="#">ExecutableEntityActivationReason.bitPosition</a>
[constr_1227]	Value of attribute <a href="#">ExecutableEntityActivationReason.bitPosition</a> shall be unique
[constr_1228]	<a href="#">RTEEvent</a> that is referenced by a <a href="#">WaitPoint</a> in the role <a href="#">trigger</a> shall not reference <a href="#">ExecutableEntityActivationReason</a>
[constr_1229]	<a href="#">category</a> of <a href="#">ImplementationDataType</a> boils down to <a href="#">VALUE</a>
[constr_1230]	<a href="#">ApplicationDataType</a> that qualifies for <a href="#">Integral Primitive Type</a>
[constr_1231]	<a href="#">ConsistencyNeeds</a> aggregated by <a href="#">CompositionSwComponentType</a>
[constr_1232]	<a href="#">ConsistencyNeeds</a> aggregated by <a href="#">AtomicSwComponentType</a>
[constr_1233]	<a href="#">InstantiationTimingEventProps</a> shall only reference <a href="#">TimingEvent</a>
[constr_1234]	Value of <a href="#">RunnableEntity.symbol</a>
[constr_1237]	Scope of mapped <a href="#">ClientServerOperations</a> in the context of a <a href="#">ClientServerOperationMapping</a>
[constr_1238]	Scope of mapped <a href="#">ApplicationErrors</a> in the context of a <a href="#">ClientServerOperationMapping</a>
[constr_1240]	Consistency of <a href="#">ArgumentDataPrototypes</a> within the context of a <a href="#">ClientServerOperationMapping</a>





Number	Heading
[constr_1241]	Compound Primitive Data Types and <code>invalidValue</code>
[constr_1242]	Restriction of <code>invalidValue</code> for <code>ApplicationPrimitiveDataType</code> of category <code>STRING</code>
[constr_1243]	<code>NumericalOrText</code> shall either define <code>vf</code> or <code>vt</code>
[constr_1244]	<code>DataPrototypes</code> used in application software shall not be typed by C enums
[constr_1245]	Consideration of <code>ModeTransitions</code> for the compatibility of <code>ModeDeclarationGroups</code>
[constr_1246]	Consistency of <code>firstMode</code> and <code>secondMode</code> in the scope of one <code>ModeDeclarationMappingSet</code>
[constr_1247]	Consistency of <code>ModeDeclarationMappingSet</code> with respect to the referenced <code>firstModeGroup</code> and <code>secondModeGroup</code>
[constr_1248]	Compatibility of <code>PortPrototypes</code> of different <code>DataInterfaces</code> in the context of a <code>PassThroughSwConnector</code>
[constr_1249]	Compatibility of <code>ModeSwitchInterfaces</code> in the context of a <code>PassThroughSwConnector</code>
[constr_1250]	Compatibility of <code>ClientServerInterfaces</code> in the context of a <code>PassThroughSwConnector</code>
[constr_1251]	Compatibility of <code>PortPrototypes</code> of <code>TriggerInterfaces</code> in the context of a <code>PassThroughSwConnector</code>
[constr_1252]	Creation of a loop involving a <code>PassThroughSwConnector</code> is not allowed
[constr_1253]	Supported usage of <code>VariationPointProxy</code>
[constr_1254]	Definition of a pointer to a pointer
[constr_1255]	<code>ApplicationPrimitiveDataTypes</code> of category <code>BOOLEAN</code> and <code>STRING</code>
[constr_1256]	Acknowledgement feedback in n:1 writer case
[constr_1257]	No <code>WaitPoints</code> allowed
[constr_1258]	Value of <code>minimumStartInterval</code> for <code>RunnableEntities</code> triggered by an <code>InitEvent</code>
[constr_1259]	Aggregation of <code>AsynchronousServerCallPoint</code> and <code>AsynchronousServerCallResultPoint</code>
[constr_1260]	No mode disabling for <code>InitEvents</code>
[constr_1261]	Applicability for <code>EndToEndDescription.dataIdNibbleOffset</code>
[constr_1263]	Existence of <code>ModeErrorBehavior.defaultMode</code>
[constr_1264]	Iteration along output axis is only supported for <code>VALUE</code> and <code>VAL_BLK</code>
[constr_1268]	<code>ArgumentDataPrototype.direction</code> shall be preserved in a <code>ClientServerOperationMapping</code>
[constr_1269]	Number of <code>arguments</code> shall be preserved in a <code>ClientServerOperationMapping</code>
[constr_1270]	<code>ArgumentDataPrototype</code> shall be mapped only once in a <code>ClientServerOperationMapping</code>
[constr_1271]	<code>RecordValueSpecification.fields</code> shall be identical to the number of <code>ApplicationRecordDataType.elements</code>
[constr_1272]	<code>RecordValueSpecification.fields</code> shall be identical to the number of <code>subElements</code> of <code>ImplementationDataType</code> of category <code>STRUCTURE</code>





Number	Heading
[constr_1273]	Rules for the initialization of <a href="#">ApplicationArrayDataType</a> by means of <a href="#">ArrayValueSpecification</a>
[constr_1274]	Rules for the initialization of array-shaped <a href="#">ImplementationDataType</a> with a fixed size by means of <a href="#">ArrayValueSpecification</a>
[constr_1277]	<a href="#">SwDataDefProps.swImplPolicy</a> of a <a href="#">VariableDataPrototype</a> referenced by a <a href="#">VariableAccess</a> aggregated in the role <a href="#">dataReceivePointByValue</a>
[constr_1278]	<a href="#">PhysConstrs</a> references a <a href="#">Unit</a>
[constr_1279]	Unmapped elements of <a href="#">ApplicationCompositeDataTypes</a> or <a href="#">ImplementationDataTypes</a> and the attribute <a href="#">swImplPolicy</a>
[constr_1280]	Unmapped <a href="#">dataElement</a> on the “target” end shall have an <a href="#">initValue</a>
[constr_1282]	Restriction concerning the usage of <a href="#">RuleBasedValueSpecification</a> or a <a href="#">ReferenceValueSpecification</a> for the specification of an <a href="#">invalidValue</a>
[constr_1284]	Limitation of the use of <a href="#">TextValueSpecification</a>
[constr_1285]	Applicability of roles vs. <a href="#">PortPrototypes</a>
[constr_1286]	<a href="#">serverArgumentImplPolicy</a> and <a href="#">ArgumentDataPrototype</a> typed by primitive data types
[constr_1287]	Compatibility of <a href="#">SenderReceiverInterfaces</a> with respect to <a href="#">invalidationPolicy</a>
[constr_1288]	Allowed Attributes vs. <a href="#">category</a> for <a href="#">DataPrototypes</a> typed by <a href="#">ImplementationDataTypes</a>
[constr_1289]	Allowed Attributes vs. <a href="#">category</a> for <a href="#">DataPrototypes</a> typed by <a href="#">ApplicationDataTypes</a>
[constr_1290]	Limitation on the number of <a href="#">PPortComSpecs</a> in the context of one <a href="#">PPortPrototype</a>
[constr_1291]	Limitation on the number of <a href="#">RPortComSpecs</a> in the context of one <a href="#">PPortPrototype</a>
[constr_1292]	Limitation on the number of <a href="#">RPortComSpecs</a> / <a href="#">PPortComSpecs</a> in the context of one <a href="#">PRPortPrototype</a>
[constr_1295]	<a href="#">PortInterfaces</a> and <a href="#">category</a> <a href="#">DATA_REFERENCE</a>
[constr_1296]	<a href="#">DataPrototypes</a> used as <a href="#">explicitInterRunnableVariable</a> or <a href="#">implicitInterRunnableVariable</a> and <a href="#">category</a> <a href="#">DATA_REFERENCE</a>
[constr_1298]	Existence of attributes if <a href="#">category</a> of a <a href="#">ModeDeclarationGroup</a> is set to <a href="#">EXPLICIT_ORDER</a>
[constr_1299]	Existence of attributes if <a href="#">category</a> of a <a href="#">ModeDeclarationGroup</a> is set to other than <a href="#">EXPLICIT_ORDER</a>
[constr_1300]	Primitive <a href="#">DataPrototype</a> on the “source” end shall not be mapped to element of a composite data type on the “target” end of the <a href="#">SwConnector</a>
[constr_1301]	Existence of <a href="#">RoleBasedDataTypeAssignment.role</a> vs. <a href="#">RoleBasedDataAssignment.role</a>
[constr_1302]	Restriction of data invalidation
[constr_1303]	Applicability of <a href="#">TextTableMapping</a> depending on the value of <a href="#">CompuMethod.category</a>
[constr_1304]	Existence of attribute <a href="#">bitfieldTextTableMaskFirst</a>





Number	Heading
[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>
[constr_1312]	<code>PortPrototypes</code> typed by a <code>ParameterInterface</code>
[constr_1313]	Completeness of <code>TextTableMapping</code> for the values of a given bit mask on the sender side
[constr_1314]	Profile <code>VSA_LINEAR</code> for <code>ApplicationArrayDataType</code>
[constr_1315]	Profile <code>VSA_SQUARE</code> for <code>ApplicationArrayDataType</code>
[constr_1316]	Profile <code>VSA_RECTANGULAR</code> for <code>ApplicationArrayDataType</code>
[constr_1317]	Profile <code>VSA_FULLY_FLEXIBLE</code> for <code>ApplicationArrayDataType</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 <code>VSA_FULLY_FLEXIBLE</code> for <code>ImplementationDataType</code>
[constr_1322]	<code>Size Indicator</code> for undefined <code>dynamicArraySizeProfile</code>
[constr_1363]	Existence of attributes of <code>DiagnosticValueNeeds</code>
[constr_1364]	Existence of attributes of <code>DiagnosticIoControlNeeds</code>
[constr_1375]	Existence of attributes of <code>CompuMethod</code> and related meta-classes
[constr_1381]	Appearance of core-related possible values of <code>MemorySection.option</code> or <code>SwAddrMethod.option</code>
[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>





Number	Heading
[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>NvBlockDataMapping</code> s 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>
[constr_1407]	Definition of <code>SwDataDefProps.dataConstr</code> depending on the capabilities of the data type
[constr_1408]	Definition of <code>SwDataDefProps.displayFormat</code> depending on the capabilities of the data type
[constr_1409]	Definition of <code>SwDataDefProps.dataConstr</code> depending on the capabilities of the element data type
[constr_1410]	Definition of <code>SwDataDefProps.displayFormat</code> depending on the capabilities of the element data type
[constr_1413]	Definition of <code>SwDataDefProps.stepSize</code> depending on the capabilities of the data type
[constr_1414]	Definition of <code>SwDataDefProps.stepSize</code> depending on the capabilities of the element data type
[constr_1415]	Supported values of <code>ModeSwitchEventTriggeredActivity.role</code>
[constr_1416]	Existence of <code>ApplicationArrayElement.maxNumberOfElements</code>
[constr_1417]	Invalid connection between <code>NvBlockSwComponentType</code> and other <code>AtomicSwComponentType</code> (I)
[constr_1418]	Invalid connection between <code>NvBlockSwComponentType</code> and other <code>AtomicSwComponentType</code> (II)
[constr_1420]	Existence of <code>SwAxisIndividual.inputVariableType</code>
[constr_1422]	Value of <code>category</code> is <code>VOID</code>
[constr_1423]	Completeness of references <code>ArVariableInImplementationDataInstanceRef.contextDataPrototype</code>





Number	Heading
[constr_1424]	Existence of <code>ArVariableInImplementationDataInstanceRef</code> . <code>contextDataPrototype</code>
[constr_1425]	Definition of <code>swCalprmAxisSet.swCalprmAxis / SwAxisIndividual</code> . <code>swVariableRef</code> depending on the capabilities of the data type
[constr_1426]	Consistency of array sizes for axes and input variable array
[constr_1427]	Definition of <code>swCalprmAxisSet.swCalprmAxis / SwAxisGrouped</code> . <code>swCalprmRef</code> depending on the capabilities of the data type
[constr_1428]	Consistency of array sizes for arrays of elements of <code>category CURVE, MAP, CUBOID, CUBE_4, or CUBE_5</code> arrays and used group axes arrays
[constr_1429]	Access to data within <code>PortPrototypes</code> from within <code>RunnableEntitys</code>
[constr_1430]	Access to local data from within <code>RunnableEntitys</code>
[constr_1431]	Access to parameters from within <code>RunnableEntitys</code>
[constr_1432]	Multiplicity of <code>CommunicationBufferLocking</code>
[constr_1433]	Transient faults are not applicable to software-components
[constr_1434]	<code>CompuScales</code> shall not have identical <code>CompuScale Value Symbolic Names</code>
[constr_1438]	<code>ApplicationArrayElement.indexDataType</code> needs to refer to a <code>CompuMethod</code> of <code>category TEXTTABLE</code>
[constr_1439]	Requirements on <code>ApplicationArrayElement</code> if attribute <code>indexDataType</code> exists
[constr_1440]	Size of the <code>CompuMethod</code> of <code>category TEXTTABLE</code> referenced by <code>ApplicationArrayElement.indexDataType</code>
[constr_1442]	<code>category TYPE_REFERENCE</code> shall not be used for modeling the “payload” of a <code>Wrapped Union Data Type</code>
[constr_1444]	Limited applicability of <code>Wrapped Union Data Type</code>
[constr_1445]	Initialization of the <code>Member Selector</code> of a <code>Wrapped Union Data Type</code>
[constr_1446]	No definition of <code>invalidValue</code> for a <code>Wrapped Union Data Type</code>
[constr_1468]	Limitation on the number of <code>SwcExclusiveAreaPolicys</code>
[constr_1469]	Applicability of constraints depending on the existence of a data transformation
[constr_1516]	Completeness of references <code>ArParameterInImplementationDataInstanceRef</code> . <code>contextDataPrototype</code>
[constr_1517]	Existence of <code>ArParameterInImplementationDataInstanceRef</code> . <code>contextDataPrototype</code>
[constr_1518]	Consistency of data types in the context of <code>ArParameterInImplementationDataInstanceRef</code>
[constr_1519]	Existence of attributes vs. <code>category</code> of <code>ApplicationValueSpecification</code>
[constr_1520]	Semantics of <code>ObdRatioServiceNeeds.rateBasedMonitoredEvent</code>
[constr_1521]	Reference from <code>AsynchronousServerCallReturnsEvent</code> to <code>AsynchronousServerCallResultPoint</code>
[constr_1523]	No mode disabling for <code>OperationInvokedEvents</code>
[constr_1538]	Restriction for <code>ReceiverComSpec.dataElement</code>
[constr_1539]	Restriction for <code>SenderComSpec.dataElement</code>





Number	Heading
[constr_1540]	Existence of <code>ClientComSpec.operation</code>
[constr_1541]	Existence of <code>ServerComSpec.operation</code>
[constr_1544]	Modeling of <code>SwAxisGeneric</code> for the definition of a fix axis
[constr_1545]	No initialization for fix axis
[constr_1583]	<code>PortInterfaceMapping</code> for <code>DataPrototype</code> typed by <code>Compound Primitive Data Type</code>
[constr_1592]	Definition of <code>SwDataDefProps.displayPresentation</code> depending on the capabilities of the data type
[constr_1602]	Definition of <code>SwDataDefProps.displayPresentation</code> depending on the capabilities of the element
[constr_1607]	Only <code>Wrapped Union Data Types</code> in <code>PortInterface</code>
[constr_1608]	Existence of <code>rootParameterDataPrototype</code>
[constr_1609]	Existence of <code>rootVariableDataPrototype</code>
[constr_1610]	Existence of <code>SwDataDefProps.swValueBlockSize</code> and <code>SwDataDefProps.swValueBlockSizeMult</code>
[constr_1622]	Value of <code>TimingEvent.offset</code> vs. <code>TimingEvent.period</code>
[constr_1631]	Applicability of <code>DataPrototypeMapping.secondToFirstDataTransformation</code>
[constr_1632]	Restriction for <code>firstToSecondDataTransformation</code> and <code>secondToFirstDataTransformation</code>
[constr_1634]	Allowed combinations of <code>ApplicationDataType.category</code> vs. <code>CompuMethod.category</code>
[constr_1635]	Relevance of attribute <code>isOptional</code>
[constr_1636]	Mapping of data types that represent an <code>Optional Element Structure</code>
[constr_1637]	Existence of <code>ImplementationDataTypeElement.isOptional</code> vs. <code>ImplementationDataType.isStructWithOptionalElement</code>
[constr_1638]	First <code>ImplementationDataTypeElement</code> of <code>ImplementationDataType</code> that represents an <code>Optional Element Structure</code>
[constr_1639]	<code>ImplementationDataTypeElement</code> with attribute <code>isOptional</code> set to <code>True</code>
[constr_1640]	No use of <code>Optional Element Structure</code> for interaction with the diagnostic stack
[constr_1662]	Compatibility of <code>ApplicationRecordDataType</code> and <code>ImplementationDataType</code> that both represent an <code>Optional Element Structure</code>
[constr_1679]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = signalBasedDiagnostics</code>
[constr_1680]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = AppModeRequestInterface</code>
[constr_1681]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = VerificationStatus</code>







Number	Heading
[constr_1682]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xFacVdp</code>
[constr_1683]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xApp1RxIndicationCam</code>
[constr_1684]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xApp1RxIndicationMapem</code>
[constr_1685]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xApp1RxIndicationIvim</code>
[constr_1686]	Existence of attribute <code>RoleBasedDataAssignment.usedDataElement.localVariable</code> for <code>RoleBasedDataAssignment.role = V2xApp1RxIndicationSpatem</code>
[constr_1694]	Allowed target of <code>SwDataDefProps.implementationDataType</code>
[constr_1706]	Definition of initial value for data transmission
[constr_1712]	Existence of attribute <code>ArrayValueSpecification.intendedPartialInitializationCount</code>
[constr_1713]	<code>NvBlockDescriptor.writingStrategy.usedDataElement</code> shall refer to <code>AutosarDataPrototype</code>
[constr_1714]	<code>AutosarDataPrototype</code> shall only be referenced by a single <code>NvBlockDescriptor.writingStrategy</code>
[constr_1715]	Possible values of attribute <code>NvBlockDescriptor.writingStrategy.role</code>
[constr_1716]	Consistency of attribute <code>NvBlockDescriptor.writingStrategy.role</code> set to <code>storeAtShutdown</code>
[constr_1717]	Consistency of attribute <code>NvBlockDescriptor.writingStrategy.role</code> set to <code>storeImmediate</code>
[constr_1718]	Inheritance of <code>SwDataDefProps.dataConstr</code> from an array data type to the array elements
[constr_1719]	Inheritance of <code>SwDataDefProps.displayFormat</code> from an array data type to the array elements
[constr_1720]	Inheritance of <code>SwDataDefProps.stepSize</code> from an array data type to the array elements
[constr_1724]	Usage of attribute <code>ClientServerOperation.diagArgIntegrity</code>
[constr_1726]	Ordering of <code>MetaDataItemSet.metaDataItem</code>
[constr_1735]	Limitation of the aggregation of <code>AutosarVariableRef</code> in the context of an <code>NvBlockDataMapping</code> owned by a <code>BulkNvDataDescriptor</code>
[constr_1741]	Restriction to explicit sending semantics for the usage of <code>DataServices</code> in the context of a <code>SwcServiceDependency</code> that aggregates <code>DiagnosticValueNeeds</code> that in turn is referenced by a <code>DiagnosticIoControlNeeds</code>
[constr_1754]	Aggregation of <code>NumericalRuleBasedValueSpecification</code>
[constr_1755]	Aggregation of <code>CompositeRuleBasedValueSpecification</code>
[constr_1771]	Existence of <code>SwValueCont.unit</code>
[constr_1773]	Value of attribute <code>dataSendPoint.returnValueProvision</code>





Number	Heading
[constr_1774]	Value of attribute <a href="#">dataReceivePointByArgument.returnValueProvision</a>
[constr_1775]	Value of attribute <a href="#">serverCallPoint.returnValueProvision</a>
[constr_1776]	Value of attribute <a href="#">asynchronousServerCallResultPoint.returnValueProvision</a>
[constr_1777]	Value of attribute <a href="#">externalTriggeringPoint.returnValueProvision</a>
[constr_1778]	Value of attribute <a href="#">modeSwitchPoint.returnValueProvision</a>
[constr_1779]	Scope of the definition of an <a href="#">AbstractRuleBasedValueSpecification</a>
[constr_1783]	Existence of attribute <a href="#">ImplementationDataTypeElement.arrayImplPolicy</a>
[constr_1860]	Multiplicity of <a href="#">DelegationSwConnector.innerPort</a>
[constr_1861]	Multiplicity of <a href="#">DelegationSwConnector.outerPort</a>
[constr_1862]	Multiplicity of <a href="#">PassThroughSwConnector.requiredOuterPort</a>
[constr_1863]	Multiplicity of <a href="#">PassThroughSwConnector.providedOuterPort</a>
[constr_1864]	Multiplicity of <a href="#">InstantiationRTEEventProps.refinedEvent</a>
[constr_1865]	Existence of <a href="#">InvalidationPolicy.dataElement</a>
[constr_1866]	Existence of <a href="#">MetaDataItem.length</a>
[constr_1867]	Existence of <a href="#">MetaDataItem.metaDataType</a>
[constr_1868]	Existence of <a href="#">MetaDataItemSet.dataElement</a>
[constr_1869]	Existence of attribute <a href="#">ArgumentDataPrototype.direction</a>
[constr_1870]	Existence of attribute <a href="#">ApplicationError.errorCode</a>
[constr_1871]	Existence of attribute <a href="#">ModeRequestTypeMap.implementationDataType</a>
[constr_1872]	Existence of attribute <a href="#">ModeRequestTypeMap.modeGroup</a>
[constr_1888]	Existence of attribute <a href="#">DataTransformation.executeDespiteDataUnavailability</a>
[constr_1889]	Existence of attribute <a href="#">QueuedReceiverComSpec.queueLength</a>
[constr_1891]	Existence of attribute <a href="#">NonqueuedReceiverComSpec.initValue</a>
[constr_1892]	Existence of attribute <a href="#">TransmissionAcknowledgementRequest.timeout</a>
[constr_1895]	Existence of attribute <a href="#">ModeSwitchSenderComSpec.modeGroup</a>
[constr_1896]	Existence of attribute <a href="#">ModeSwitchReceiverComSpec.modeGroup</a>
[constr_1897]	Existence of reference <a href="#">ParameterProvideComSpec.parameter</a>
[constr_1898]	Existence of reference <a href="#">ParameterRequireComSpec.parameter</a>
[constr_1899]	Existence of reference <a href="#">NvRequireComSpec.variable</a>
[constr_1900]	Existence of reference <a href="#">NvProvideComSpec.variable</a>
[constr_1901]	Existence of attribute <a href="#">EndToEndDescription.category</a>
[constr_1902]	Existence of attribute <a href="#">EndToEndProtection.endToEndProfile</a>
[constr_1903]	Existence of reference <a href="#">DataTypeMap.applicationDataType</a>
[constr_1904]	Existence of reference <a href="#">DataTypeMap.implementationDataType</a>
[constr_1905]	Existence of attribute <a href="#">SwTextProps.arraySizeSemantics</a>
[constr_1906]	Existence of attribute <a href="#">SwTextProps.swMaxTextSize</a>
[constr_1909]	Existence of attribute <a href="#">ImplementationProps.symbol</a>
[constr_1910]	Existence of attribute <a href="#">BaseType.baseTypeDefinition</a>





Number	Heading
[constr_1911]	Existence of <code>ArVariableInImplementationDataInstanceRef.targetDataPrototype</code>
[constr_1912]	Existence of reference <code>ArParameterInImplementationDataInstanceRef.targetDataPrototype</code>
[constr_1913]	Existence of attribute <code>CompuRationalCoeffs.compuDenominator</code>
[constr_1914]	Existence of attribute <code>CompuRationalCoeffs.compuNumerator</code>
[constr_1915]	Existence of attribute <code>PhysicalDimensionMapping.firstPhysicalDimension</code>
[constr_1916]	Existence of attribute <code>PhysicalDimensionMapping.secondPhysicalDimension</code>
[constr_1917]	Existence of <code>ConstantSpecification.valueSpec</code>
[constr_1918]	Existence of <code>RecordValueSpecification.field</code>
[constr_1919]	Existence of <code>TextValueSpecification.value</code>
[constr_1920]	Existence of <code>NumericalValueSpecification.value</code>
[constr_1921]	Existence of <code>ReferenceValueSpecification.referenceValue</code>
[constr_1923]	Existence of <code>RuleBasedAxisCont.ruleBasedValues</code>
[constr_1924]	Existence of <code>RuleBasedValueCont.ruleBasedValues</code>
[constr_1925]	Existence of <code>NumericalRuleBasedValueSpecification.ruleBasedValues</code>
[constr_1926]	Existence of <code>RuleBasedValueSpecification.rule</code>
[constr_1927]	Existence of <code>RuleBasedValueSpecification.arguments</code>
[constr_1928]	Existence of <code>CompositeRuleBasedValueSpecification.rule</code>
[constr_1929]	Existence of <code>CompositeRuleBasedValueSpecification.argument</code>
[constr_1930]	Existence of <code>ConstantReference.constant</code>
[constr_1931]	Existence of <code>ConstantSpecificationMapping.applConstant</code>
[constr_1932]	Existence of <code>ConstantSpecificationMapping.implConstant</code>
[constr_1933]	Existence of <code>CalibrationParameterValue.initializedParameter</code>
[constr_1935]	Existence of attribute <code>SwcInternalBehavior.supportsMultipleInstantiation</code>
[constr_1936]	Existence of attribute <code>RunnableEntity.symbol</code>
[constr_1938]	Existence of attribute <code>RunnableEntityArgument.symbol</code>
[constr_1939]	Existence of attribute <code>ExecutableEntityActivationReason.bitPosition</code>
[constr_1940]	Existence of attribute <code>AsynchronousServerCallReturnsEvent.eventSource</code>
[constr_1941]	Existence of attribute <code>DataSendCompletedEvent.eventSource</code>
[constr_1942]	Existence of attribute <code>DataWriteCompletedEvent.eventSource</code>
[constr_1943]	Existence of attribute <code>DataReceivedEvent.data</code>
[constr_1944]	Existence of attribute <code>DataReceiveErrorEvent.data</code>
[constr_1945]	Existence of attribute <code>OperationInvokedEvent.operation</code>
[constr_1952]	Existence of reference <code>WaitPoint.trigger</code>
[constr_1954]	Existence of attribute <code>VariableAccess.accessedVariable</code>
[constr_1955]	Existence of attribute <code>ServerCallPoint.operation</code>
[constr_1956]	Existence of attribute <code>ServerCallPoint.timeout</code>





Number	Heading
[constr_1957]	Existence of attribute <a href="#">AsynchronousServerCallResultPoint.asynchronousServerCallPoint</a>
[constr_1958]	Existence of attribute <a href="#">ParameterAccess.accessedParameter</a>
[constr_1959]	Existence of attribute <a href="#">InstantiationDataDefProps.swDataDefProps</a>
[constr_1960]	Existence of attribute <a href="#">PortAPIOption.port</a>
[constr_1964]	Existence of attribute <a href="#">PerInstanceMemory.type</a>
[constr_1965]	Existence of attribute <a href="#">PerInstanceMemory.typeDefinition</a>
[constr_1973]	Existence of attribute <a href="#">ModeDeclarationGroup.initialMode</a>
[constr_1974]	Existence of attribute <a href="#">ModeDeclarationGroup.modeDeclaration</a>
[constr_2000]	Compatibility of <a href="#">ClientServerOperations</a> triggering the same <a href="#">RunnableEntity</a>
[constr_2002]	Referenced <a href="#">VariableDataPrototype</a> from <a href="#">AutosarVariableRef</a> of <a href="#">VariableAccess</a> in role <a href="#">dataReadAccess</a>
[constr_2003]	Referenced <a href="#">VariableDataPrototype</a> from <a href="#">AutosarVariableRef</a> of <a href="#">VariableAccess</a> in role <a href="#">dataWriteAccess</a>
[constr_2004]	Referenced <a href="#">VariableDataPrototype</a> from <a href="#">AutosarVariableRef</a> of <a href="#">VariableAccess</a> in role <a href="#">dataSendPoint</a>
[constr_2005]	Referenced <a href="#">VariableDataPrototype</a> from <a href="#">AutosarVariableRef</a> of <a href="#">VariableAccess</a> in role <a href="#">dataReceivePointByValue</a> or <a href="#">dataReceivePointByArgument</a>
[constr_2006]	Number of <a href="#">AsynchronousServerCallResultPoint</a> referencing to one <a href="#">AsynchronousServerCallPoint</a>
[constr_2007]	Consistency of <a href="#">typeDefinition</a> attribute
[constr_2009]	Supported kinds of <a href="#">PortPrototypes</a> of a <a href="#">NvBlockSwComponentType</a>
[constr_2010]	Connections between <a href="#">SwComponentPrototypes</a> of type <a href="#">NvBlockSwComponentType</a>
[constr_2012]	Compatibility of <a href="#">ImplementationDataTypes</a> used for <a href="#">ramBlock</a> and <a href="#">romBlock</a>
[constr_2013]	Compatibility of <a href="#">ImplementationDataTypes</a> for <a href="#">NvBlockDataMapping</a>
[constr_2015]	Limitation of <a href="#">SwcInternalBehavior</a> of a <a href="#">NvBlockSwComponentType</a>
[constr_2016]	Connections between <a href="#">SwComponentPrototypes</a> of type <a href="#">ServiceProxySwComponentType</a>
[constr_2017]	Ports of <a href="#">ServiceProxySwComponentTypes</a>
[constr_2018]	Supported remote communication of a <a href="#">ServiceProxySwComponentType</a>
[constr_2019]	<a href="#">ServiceSwComponentType</a> shall have service ports only
[constr_2020]	<a href="#">dataReadAccess</a> can not be used for queued communication
[constr_2021]	<a href="#">WaitPoint</a> referencing a <a href="#">DataReceivedEvent</a> can not be used for non-queued communication
[constr_2022]	Mutually exclusive use of <a href="#">SynchronousServerCallPoints</a> and <a href="#">AsynchronousServerCallPoints</a>
[constr_2023]	Consistency of <a href="#">timeout</a> values
[constr_2024]	<a href="#">enableTakeAddress</a> is restricted to single instantiation
[constr_2026]	Referenced <a href="#">VariableDataPrototype</a> from <a href="#">AutosarVariableRef</a> of <a href="#">VariableAccess</a> in role <a href="#">writtenLocalVariable</a> and <a href="#">readLocalVariable</a>





Number	Heading
[constr_2027]	SwcServiceDependency shall be defined for service ports only
[constr_2028]	staticMemory is restricted to single instantiation
[constr_2030]	AsynchronousServerCallResultPoint combined with WaitPoint shall belong to the same RunnableEntity
[constr_2031]	Period of TimingEvent shall be greater than 0
[constr_2033]	Timeout of DataSendCompletedEvent
[constr_2034]	SwAddrMethod referenced by RunnableEntitys, BswCalledEntitys, or BswSchedulableEntitys
[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 implicitInterRunnableVariable
[constr_2039]	swImplPolicy for VariableDataPrototype in the role explicitInterRunnableVariable
[constr_2040]	swImplPolicy for VariableDataPrototype in the role arTypedPerInstanceMemory
[constr_2041]	swImplPolicy for VariableDataPrototype in the role staticMemory
[constr_2042]	swImplPolicy for ParameterDataPrototype in ParameterInterface
[constr_2043]	swImplPolicy for ParameterDataPrototype in the role romBlock
[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_2049]	Different ModeDeclarationGroups shall have different shortNames.
[constr_2050]	Mandatory information of a SwAxisCont
[constr_2051]	Mandatory information of a SwValueCont
[constr_2052]	Values of swArraysizes and the number of values provided by swValuesPhys shall be consistent.
[constr_2053]	Consistency between role IUMPRNumerator and ObdRatioServiceNeeds.connectionType
[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_2535]	Target of an autosarParameter in AutosarParameterRef shall refer to a parameter





Number	Heading
[constr_2536]	Target of an <code>autosarVariable</code> in <code>AutosarVariableRef</code> shall refer to a variable
[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_2561]	Application of <code>DataConstrRule.constrLevel</code>
[constr_4002]	Unambiguous mapping of modes to data types
[constr_4003]	Semantics of <code>SwcModeSwitchEvent</code>
[constr_4004]	Context of <code>SenderReceiverAnnotation</code>
[constr_4005]	Context of <code>ClientServerAnnotation</code>
[constr_4006]	Context of <code>ParameterPortAnnotation</code>
[constr_4007]	Context of <code>ModePortAnnotation</code>
[constr_4008]	Context of <code>TriggerPortAnnotation</code>
[constr_4009]	Context of <code>NvDataPortAnnotation</code>
[constr_4012]	Timeout of <code>ModeSwitchedAckEvent</code>
[constr_4082]	<code>RunnableEntity.reentrancyLevel</code> shall not be set.
[constr_10005]	Existence of attribute <code>NotAvailableValueSpecification.defaultPattern</code>
[constr_10006]	Valid interval of attribute <code>NotAvailableValueSpecification.defaultPattern</code>
[constr_10009]	Aggregation of <code>ApplicationRuleBasedValueSpecification</code>
[constr_10016]	Applicability of <code>OsTaskExecutionEvent</code>
[constr_10017]	Existence of attribute <code>SwAxisCont.category</code>
[constr_10018]	Existence of attribute <code>SwAxisCont.swAxisIndex</code>
[constr_10019]	Existence of attribute <code>SwAxisCont.swValuesPhys</code>
[constr_10028]	Existence of reference stereotyped <code>&lt;&lt;isOfType&gt;&gt;</code>

**Table C.71: Changed Constraints in R21-11**

### C.14.6 Deleted Constraints in R21-11

Number	Heading
[constr_1001]	Value of <code>dataId</code> shall be unique
[constr_1008]	Applicability of <code>categorys</code> <code>STRUCTURE</code> and <code>ARRAY</code>
[constr_1027]	Types for record layouts
[constr_1030]	<code>ParameterSwComponentType</code> references <code>ConstantSpecificationMappingSet</code>
[constr_1031]	<code>NvBlockSwComponentType</code> references <code>ConstantSpecificationMappingSet</code>





Number	Heading
[constr_1143]	<code>category</code> of <code>AutosarDataType</code> shall not be extended
[constr_1157]	Applicability of constraints of <code>CompuScales</code>
[constr_1160]	Size of Compound Primitive Data Type is variant
[constr_1281]	<code>invalidValue</code> is inside the scope of the <code>compuMethod</code>
[constr_1283]	<code>invalidValue</code> is outside the scope of the <code>compuMethod</code>
[constr_4000]	Local communication of mode switches
[constr_4035]	<code>ValueSpecification</code> shall fit into data type

**Table C.72: Deleted Constraints in R21-11**

## D Modeling of InstanceRef

### D.1 Introduction

The existence of so-called `InstanceRefs` is a direct consequence of 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 subclass 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 [11].



## D.2 Modeling

### D.2.1 Components and Compositions

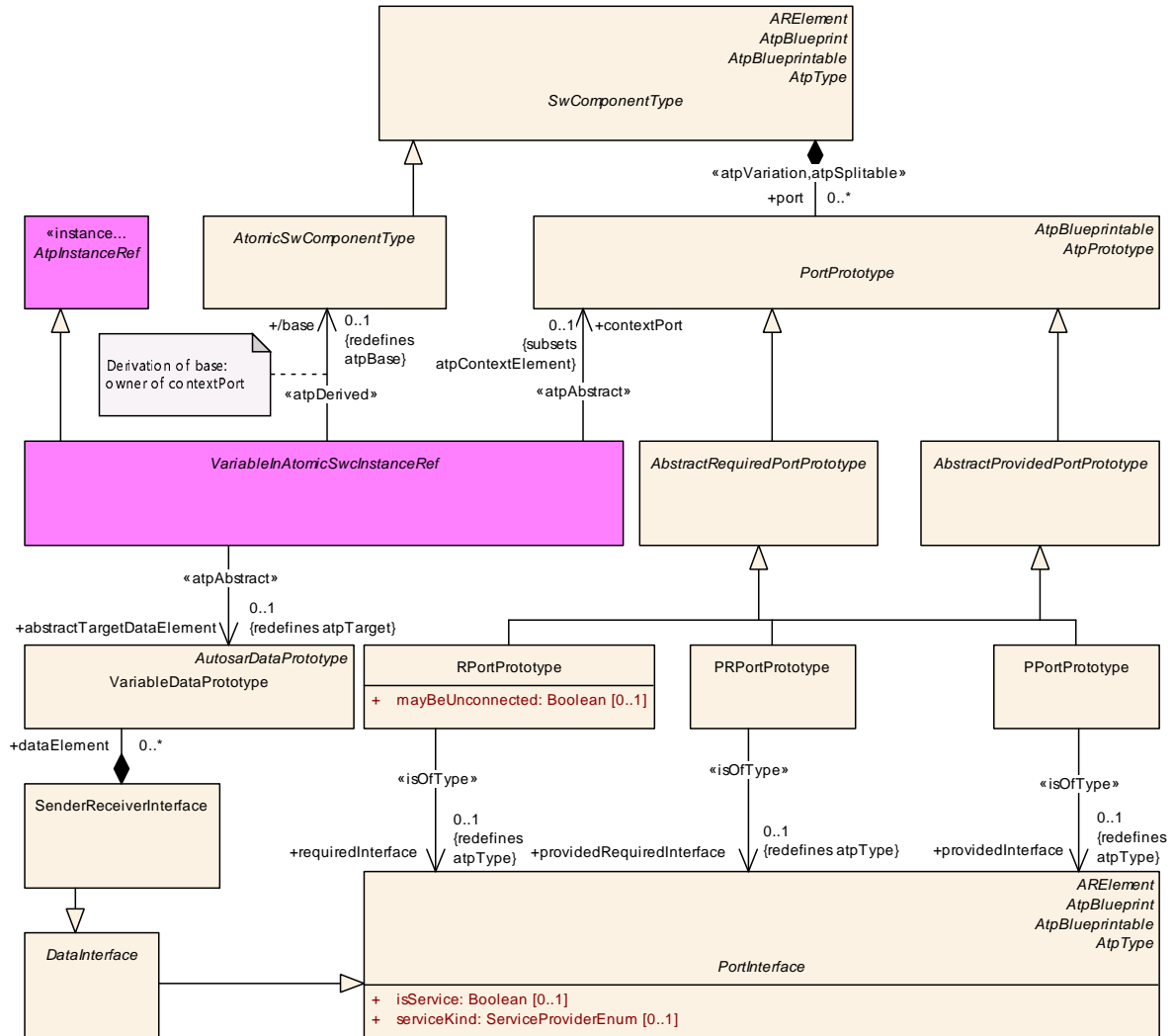


Figure D.1: Abstract modeling of references to `VariableDataPrototype` in the context of a `AtomicSwComponentType`

<b>Class</b>	<code>VariableInAtomicSwInstanceRef</code> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	<code>ARObject</code> , <code>AtpInstanceRef</code>			
<b>Subclasses</b>	<code>RVariableInAtomicSwInstanceRef</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
abstractTargetDataElement	<code>VariableDataPrototype</code>	0..1	ref	<b>Stereotypes:</b> <code>atpAbstract</code> <b>Tags:</b> <code>xml.sequenceOffset=30</code>
base	<code>AtomicSwComponentType</code>	0..1	ref	<b>Stereotypes:</b> <code>atpDerived</code> <b>Tags:</b> <code>xml.sequenceOffset=10</code>

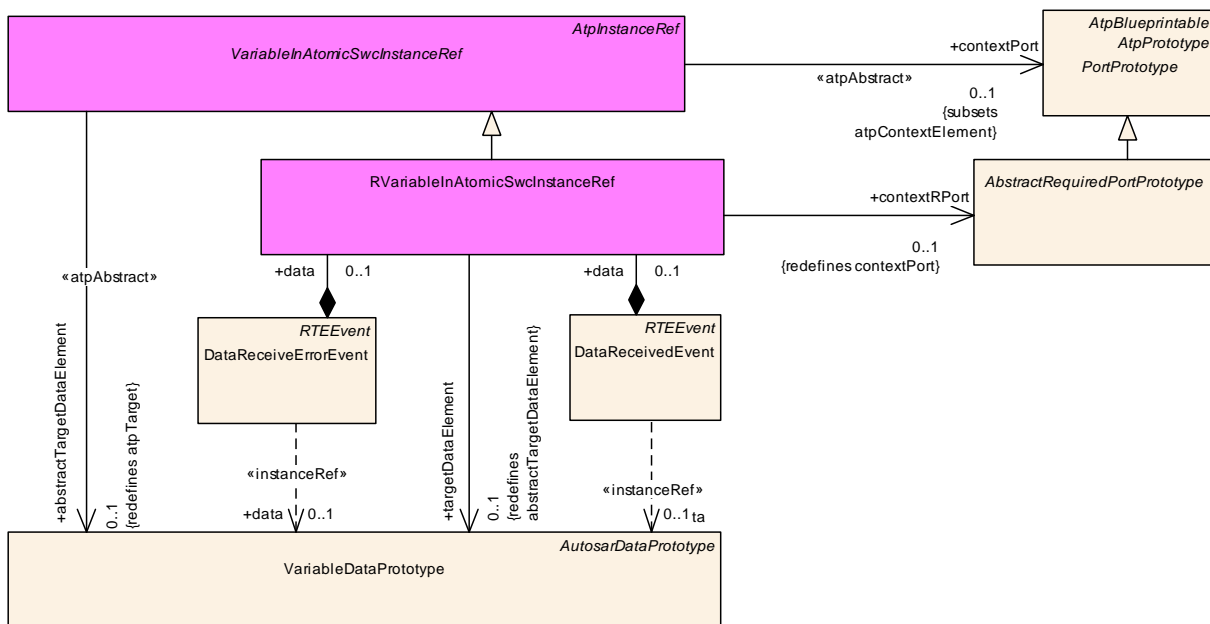




Class	VariableInAtomicSwcInstanceRef (abstract)			
contextPort	PortPrototype	0..1	ref	Stereotypes: atpAbstract Tags:xml.sequenceOffset=20

**Table D.1: VariableInAtomicSwcInstanceRef**

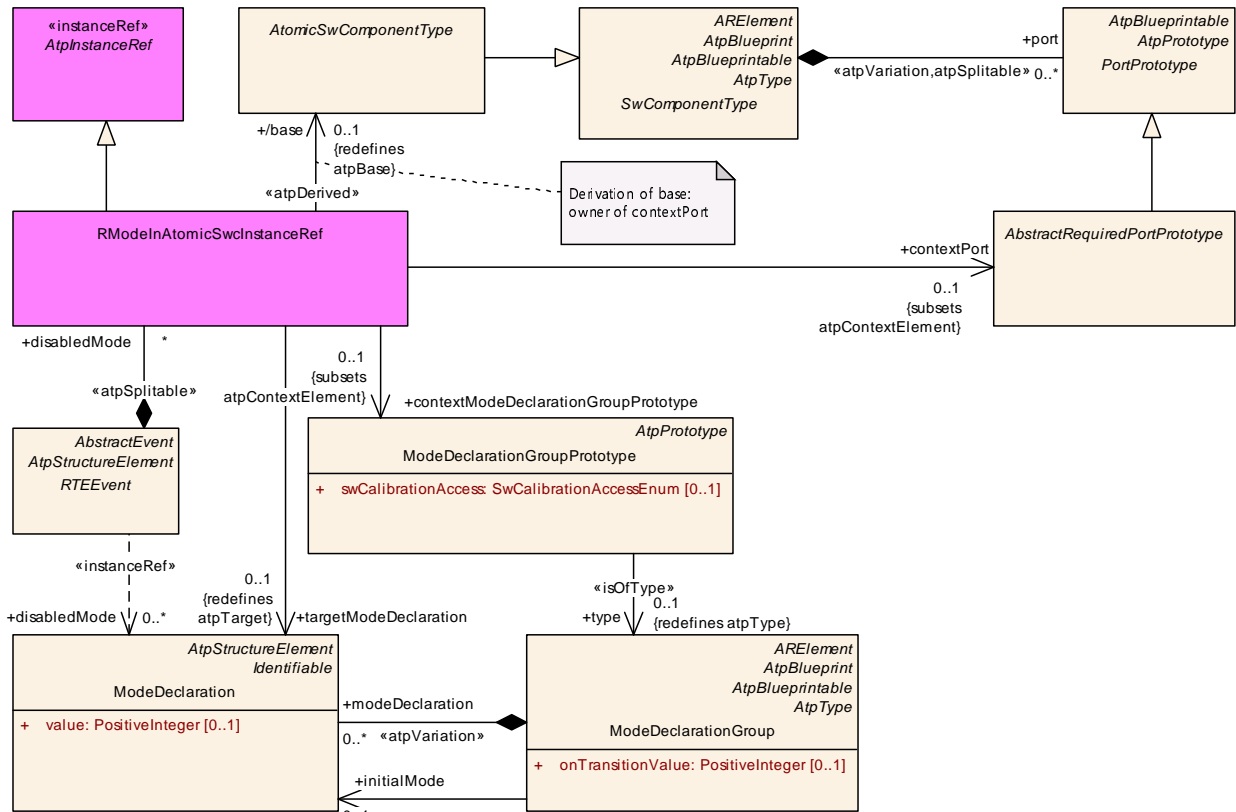
Please note the example of how the redefinition of the context association works, i.e. the association from `VariableInAtomicSwcInstanceRef` to `PortPrototype` in the role `contextPort` is **redefined** by the subclass `RVariableInAtomicSwcInstanceRef` by means of an association to `RPortPrototype` in the role `contextRPort`.



**Figure D.2: Concrete modeling of references to `VariableDataPrototype` in the context of an `RPortPrototype`**

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.

For clarification, the interpretation of the values of `xml.sequenceOffset` in this particular case is that in the generated XML Schema the `contextRPort` comes first, followed by `targetDataElement` which concludes the definition of the `InstanceRef` in the XML Schema.



**Figure D.3: Modeling of references to ModeDeclarationGroupPrototype in the context of an RPortPrototype**

<b>Class</b>	<b>RVariableInAtomicSwInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	ARObject, AtpInstanceRef, VariableInAtomicSwInstanceRef			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
contextRPort	AbstractRequiredPortPrototype	0..1	ref	Tags:xml.sequenceOffset=20
targetDataElement	VariableDataPrototype	0..1	ref	Tags:xml.sequenceOffset=30

**Table D.2: RVariableInAtomicSwInstanceRef**

<b>Class</b>	<b>RModelInAtomicSwInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	ARObject, AtpInstanceRef			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	AtomicSwComponentType	0..1	ref	Stereotypes: atpDerived Tags:xml.sequenceOffset=10
contextModeDeclarationGroupPrototype	ModeDeclarationGroupPrototype	0..1	ref	Tags:xml.sequenceOffset=30



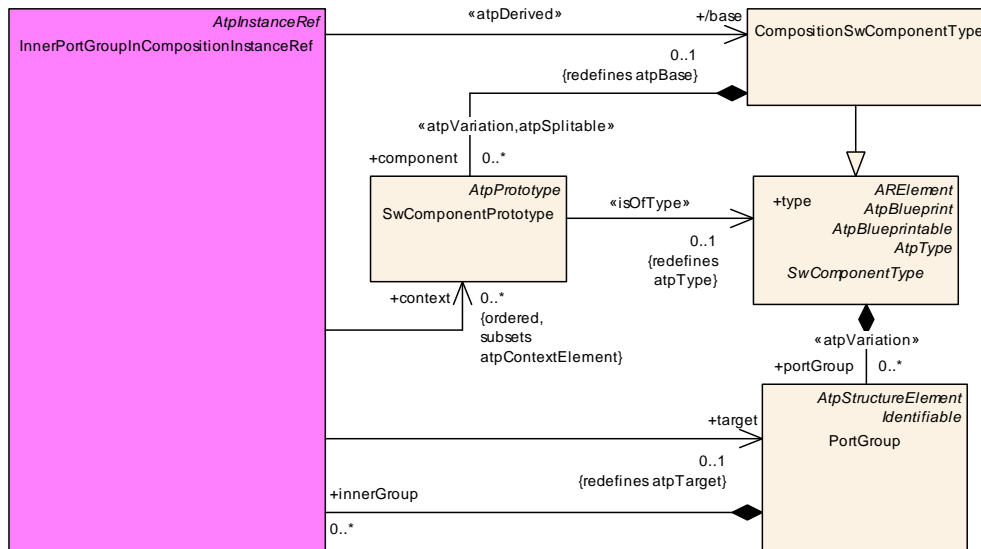


Class	RModelInAtomicSwcInstanceRef			
contextPort	<a href="#">AbstractRequiredPort Prototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=20
targetMode Declaration	<a href="#">ModeDeclaration</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=40

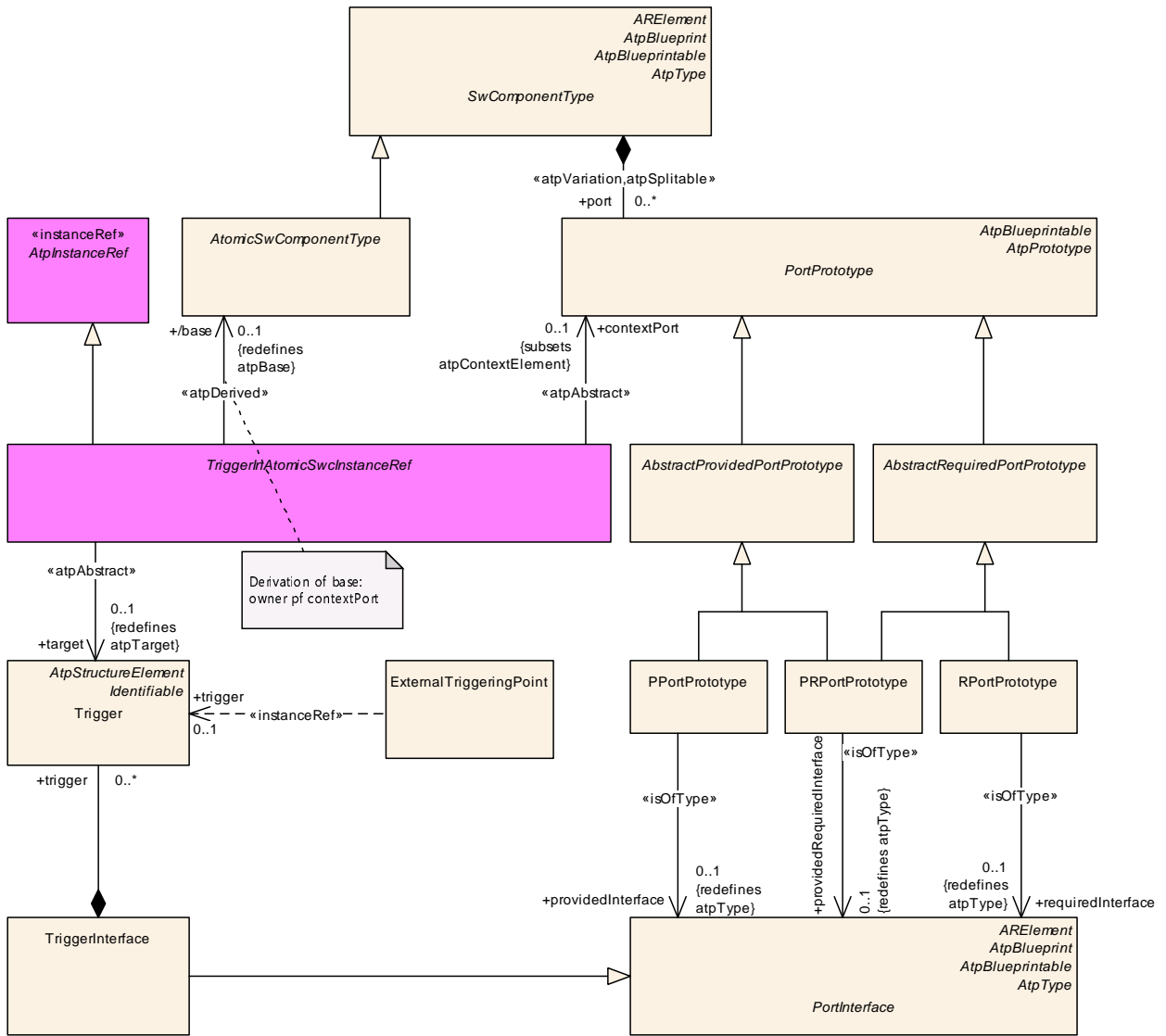
**Table D.3: RModelInAtomicSwcInstanceRef**

Class	InnerPortGroupInCompositionInstanceRef			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	<i>ARObject</i> , <i>AtpInstanceRef</i>			
Attribute	Type	Mult.	Kind	Note
base	<a href="#">CompositionSwComponentType</a>	0..1	ref	<b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
context (ordered)	<a href="#">SwComponentPrototype</a>	*	ref	<b>Tags:</b> xml.sequenceOffset=20
target	<a href="#">PortGroup</a>	0..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. <b>Tags:</b> xml.sequenceOffset=30

**Table D.4: InnerPortGroupInCompositionInstanceRef**



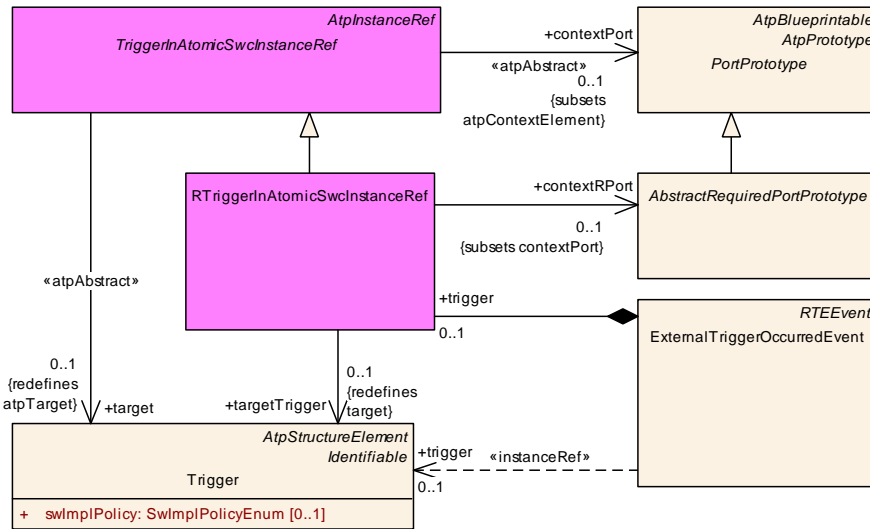
**Figure D.4: Modeling of references to [PortGroup](#) in the context of a [Composition-SwComponentType](#)**



**Figure D.5: Abstract modeling of references to **Trigger** in the context of a **Atomic-SwComponentType****

Class	<i>TriggerInAtomicSwcInstanceRef</i> (abstract)				
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs				
Note					
Base	<i>ARObject</i> , <i>AtpInstanceRef</i>				
Subclasses	<i>PTriggerInAtomicSwcTypeInstanceRef</i> , <i>RTriggerInAtomicSwcInstanceRef</i>				
Attribute	Type	Mult.	Kind	Note	
base	<a href="#">AtomicSwComponentType</a>	0..1	ref	<b>Stereotypes:</b> <i>atpDerived</i> <b>Tags:</b> xml.sequenceOffset=10	
contextPort	<a href="#">PortPrototype</a>	0..1	ref	<b>Stereotypes:</b> <i>atpAbstract</i> <b>Tags:</b> xml.sequenceOffset=20	
target	<a href="#">Trigger</a>	0..1	ref	<b>Stereotypes:</b> <i>atpAbstract</i> <b>Tags:</b> xml.sequenceOffset=30	

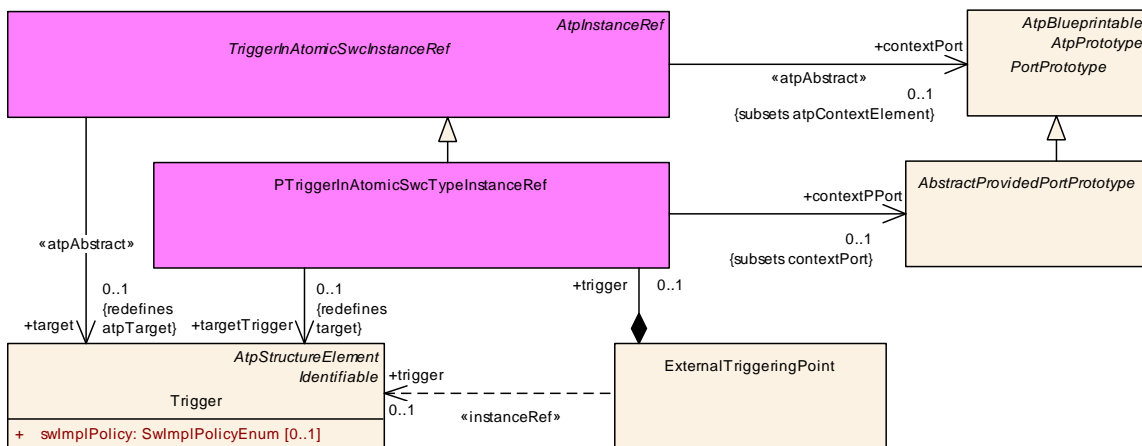
**Table D.5: TriggerInAtomicSwcInstanceRef**



**Figure D.6: Concrete modeling of references to `Trigger` in the context of an `RPortPrototype`**

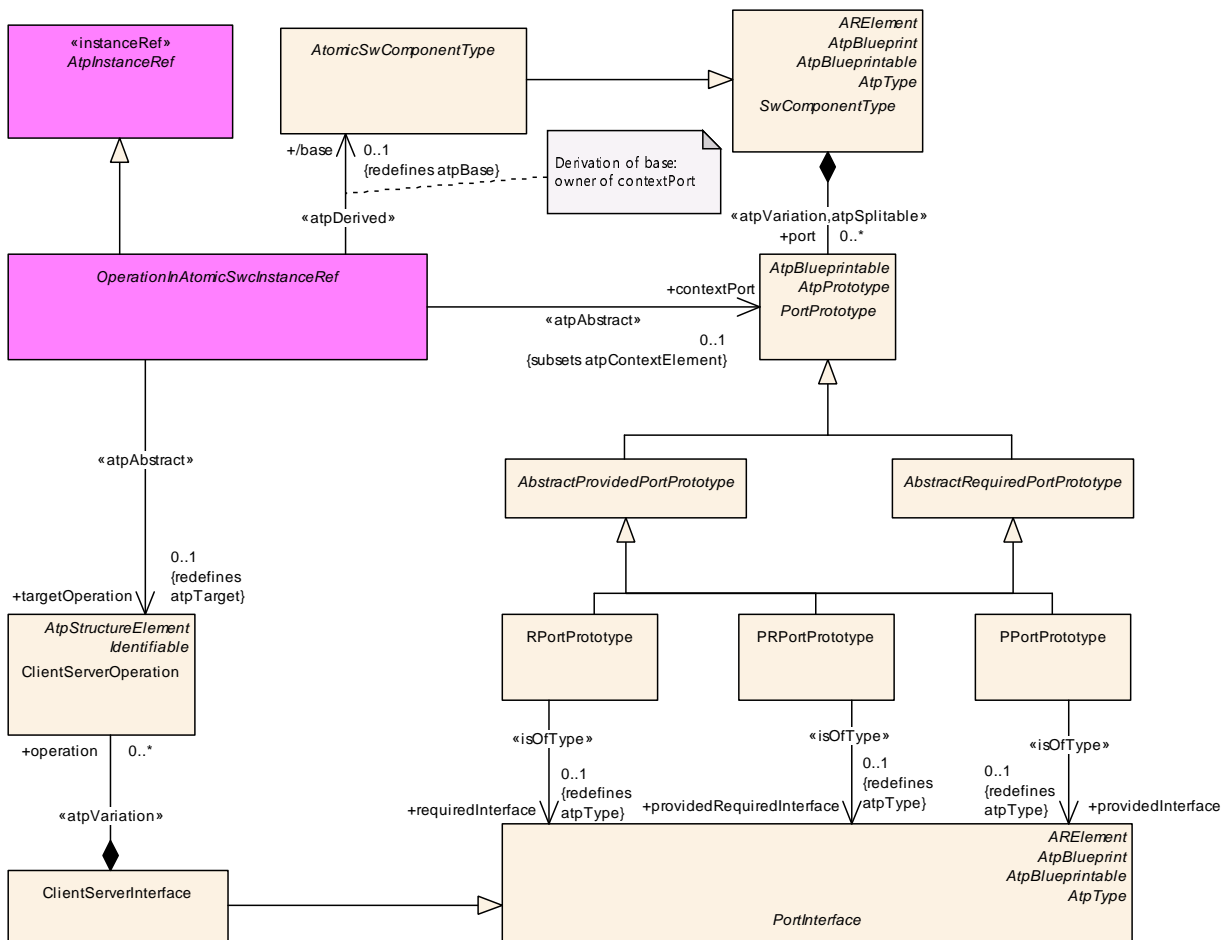
<b>Class</b>	RTriggerInAtomicSwcInstanceRef			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	ARObject, AtpInstanceRef, TriggerInAtomicSwcInstanceRef			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
contextRPort	AbstractRequiredPortPrototype	0..1	ref	Tags:xml.sequenceOffset=20
targetTrigger	Trigger	0..1	ref	Tags:xml.sequenceOffset=30

**Table D.6: RTriggerInAtomicSwcInstanceRef**



**Figure D.7: Concrete modeling of references to `Trigger` in the context of a `PPortPrototype`**

<b>Class</b>	<b>PTriggerInAtomicSwcTypeInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	ARObject, <a href="#">AtpInstanceRef</a> , <a href="#">TriggerInAtomicSwcInstanceRef</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
contextPPort	<a href="#">AbstractProvidedPortPrototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=20
targetTrigger	<a href="#">Trigger</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30

**Table D.7: PTriggerInAtomicSwcTypeInstanceRef**

**Figure D.8: Abstract modeling of references to `ClientServerOperation` in the context of a `AtomicSwComponentType`**

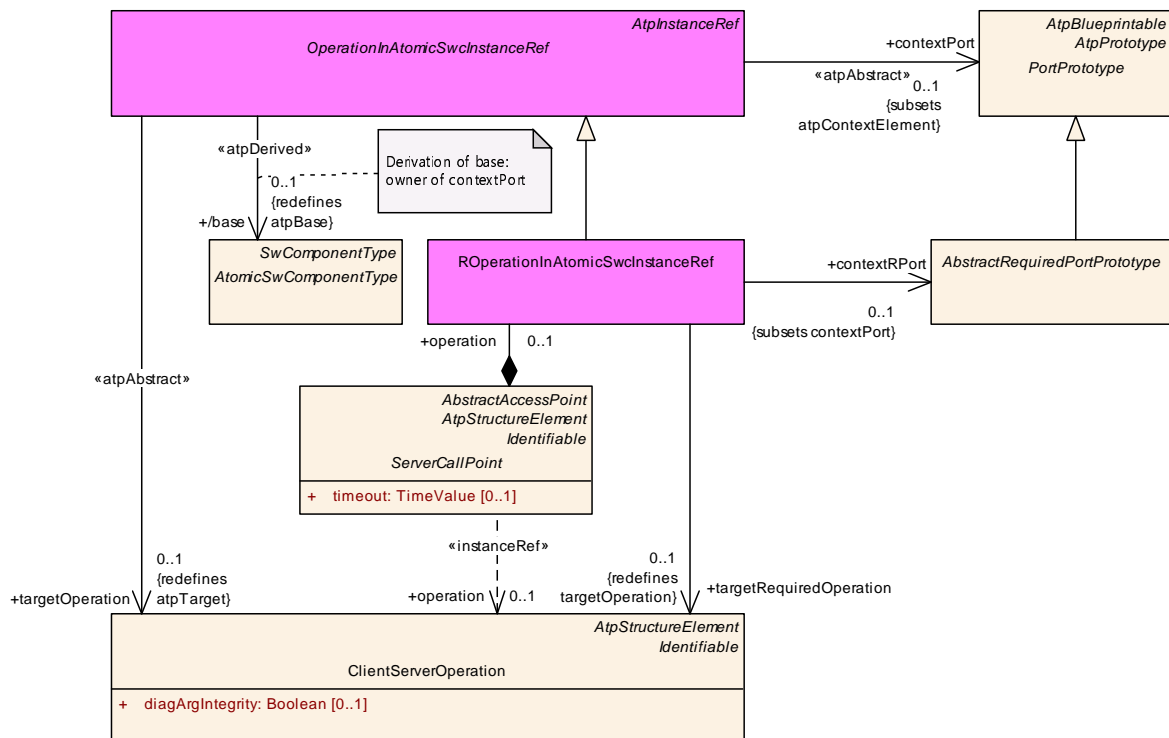
<b>Class</b>	<b>OperationInAtomicSwcInstanceRef</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	ARObject, <a href="#">AtpInstanceRef</a>			
<b>Subclasses</b>	<a href="#">POperationInAtomicSwcInstanceRef</a> , <a href="#">ROperationInAtomicSwcInstanceRef</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	<i>OperationInAtomicSwcInstanceRef</i> (abstract)			
base	<a href="#">AtomicSwComponentType</a>	0..1	ref	<b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
contextPort	<a href="#">PortPrototype</a>	0..1	ref	<b>Stereotypes:</b> atpAbstract <b>Tags:</b> xml.sequenceOffset=20
targetOperation	<a href="#">ClientServerOperation</a>	0..1	ref	<b>Stereotypes:</b> atpAbstract <b>Tags:</b> xml.sequenceOffset=30

**Table D.8: OperationInAtomicSwcInstanceRef**



**Figure D.9: Concrete modeling of references to ClientServerOperation in the context of an RPortPrototype**

Class	<b>ROperationInAtomicSwcInstanceRef</b>			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	<i>ARObject</i> , <i>AtpInstanceRef</i> , <i>OperationInAtomicSwcInstanceRef</i>			
Attribute	Type	Mult.	Kind	Note
contextRPort	<a href="#">AbstractRequiredPortPrototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=20
targetRequiredOperation	<a href="#">ClientServerOperation</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30

**Table D.9: ROperationInAtomicSwcInstanceRef**



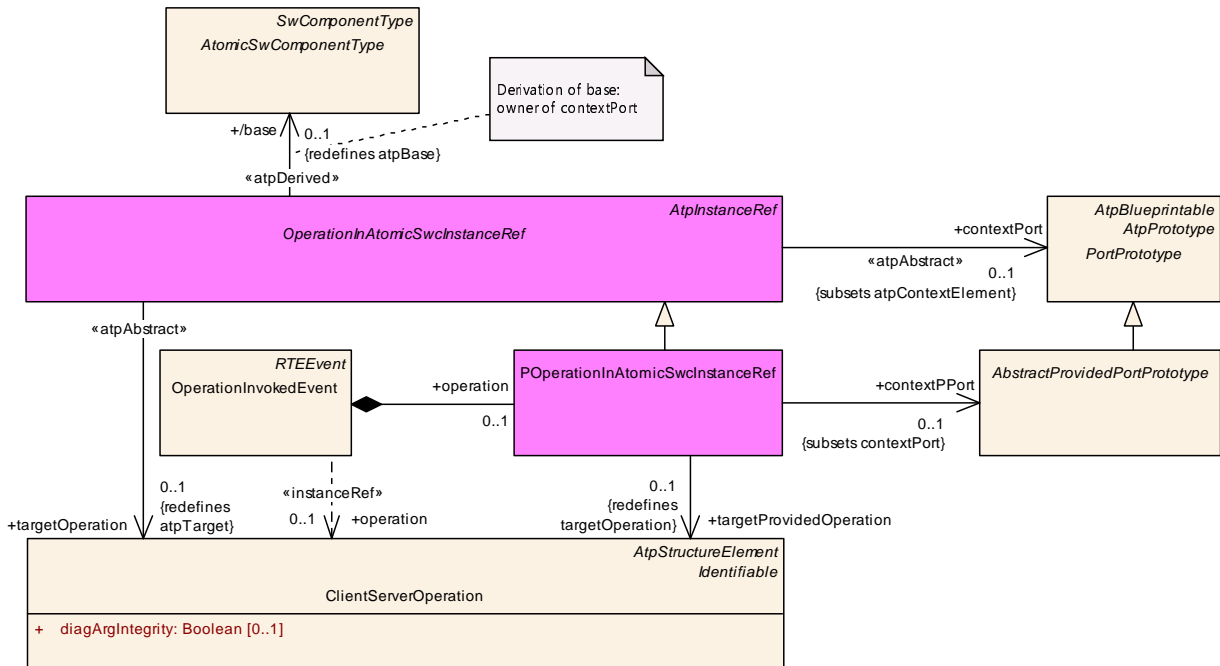


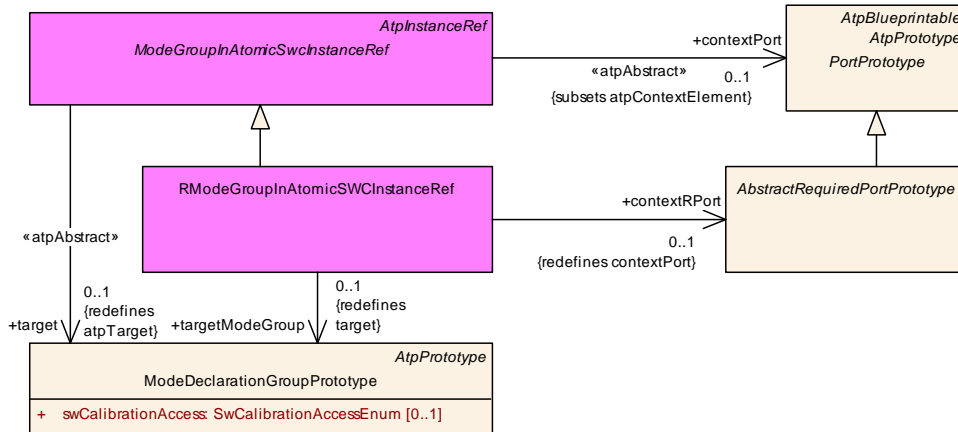
Figure D.10: Concrete modeling of references to `ClientServerOperation` in the context of a `PPortPrototype`

<b>Class</b>	<b>POperationInAtomicSwcInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	<i>ARObject</i> , <i>AtpInstanceRef</i> , <i>OperationInAtomicSwcInstanceRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
contextPPort	<a href="#">AbstractProvidedPortPrototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=20
targetProvidedOperation	<a href="#">ClientServerOperation</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30

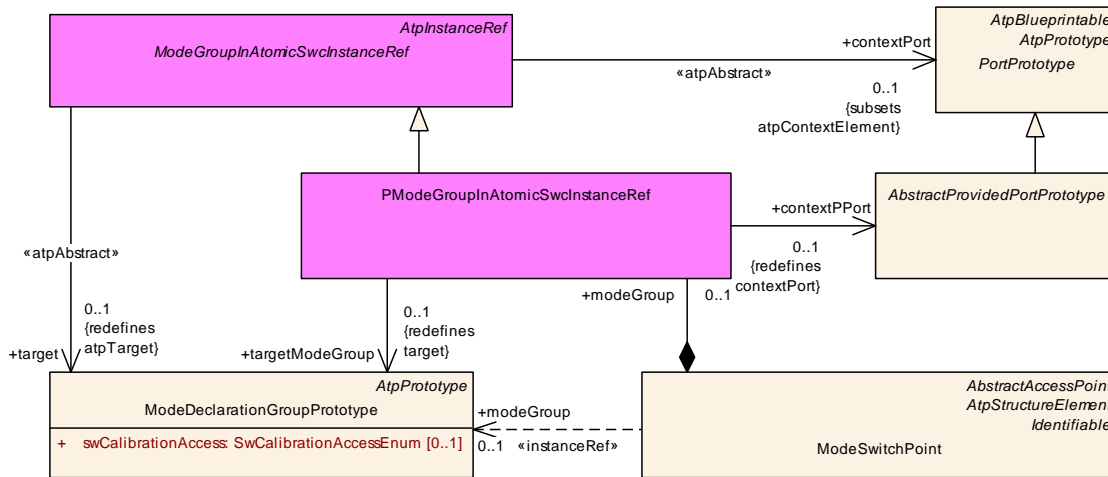
Table D.10: POperationInAtomicSwcInstanceRef

<b>Class</b>	<b>RModeGroupInAtomicSWCInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	<i>ARObject</i> , <i>AtpInstanceRef</i> , <i>ModeGroupInAtomicSwcInstanceRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
contextRPort	<a href="#">AbstractRequiredPortPrototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=20
targetModeGroup	<a href="#">ModeDeclarationGroupPrototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30

Table D.11: RModeGroupInAtomicSWCInstanceRef



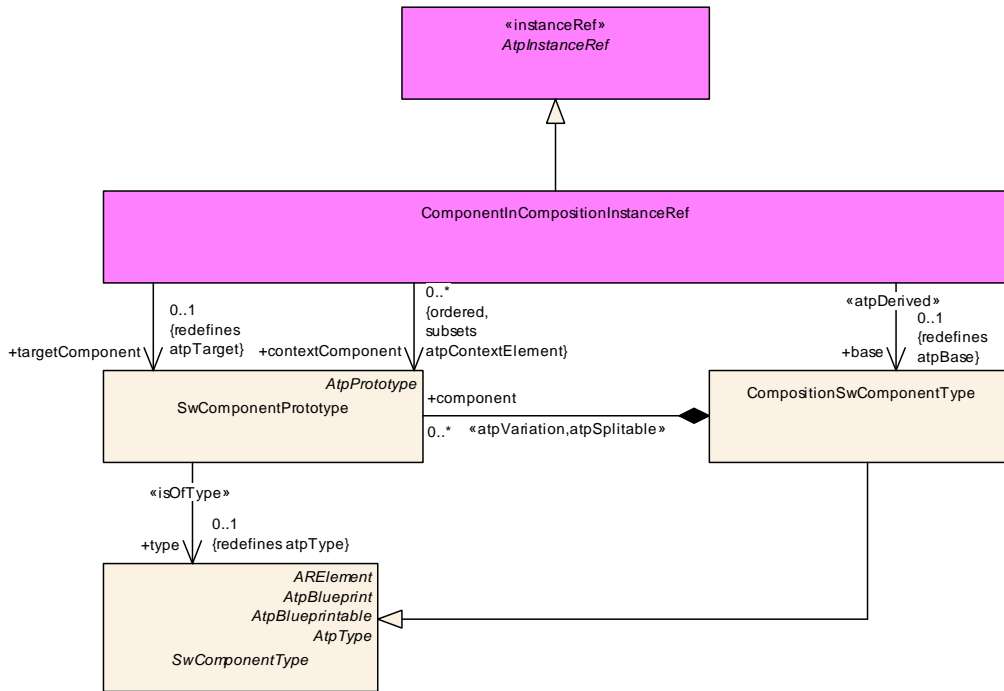
**Figure D.11: Concrete modeling of references to ModeDeclarationGroupPrototype in the context of an RPortPrototype**



**Figure D.12: Concrete modeling of references to ModeDeclarationGroupPrototype in the context of a PPortPrototype**

<b>Class</b>	<b>PModeGroupInAtomicSwcInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	ARObject, AtpInstanceRef, ModeGroupInAtomicSwcInstanceRef			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
contextPPort	AbstractProvidedPortPrototype	0..1	ref	Tags:xml.sequenceOffset=20
targetModeGroup	ModeDeclarationGroupPrototype	0..1	ref	Tags:xml.sequenceOffset=30

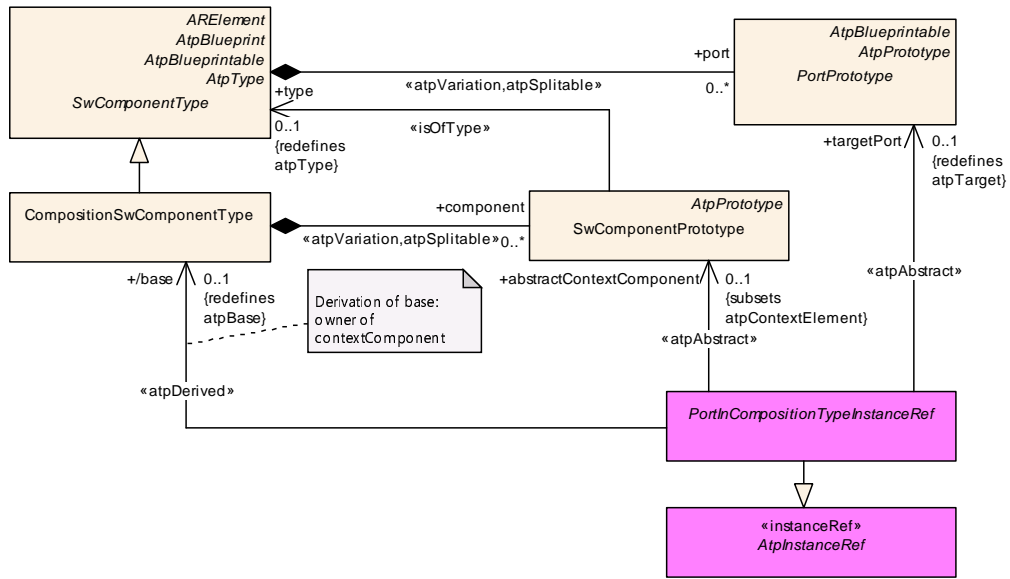
**Table D.12: PModeGroupInAtomicSwcInstanceRef**



**Figure D.13: Concrete modeling of references to *SwComponentPrototype* in the context of a *CompositionSwComponentType***

<b>Class</b>	<b>ComponentInCompositionInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
<b>Note</b>	The ComponentInCompositionInstanceRef points to a concrete SwComponentPrototype within a CompositionSwComponentType.			
<b>Base</b>	ARObject, AtpInstanceRef			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	CompositionSwComponentType	0..1	ref	<b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
context Component (ordered)	SwComponentPrototype	*	ref	The context for the scope of this timing event. <b>Tags:</b> xml.sequenceOffset=20
target Component	SwComponentPrototype	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30

**Table D.13: ComponentInCompositionInstanceRef**



**Figure D.14: Abstract modeling of references to `PortPrototype` in the context of a `CompositionSwComponentType`**

<b>Class</b>	<i>PortInCompositionTypeInstanceRef</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
<b>Note</b>				
<b>Base</b>	<i>ARObject</i> , <i>AtpInstanceRef</i>			
<b>Subclasses</b>	<i>PPortInCompositionInstanceRef</i> , <i>RPortInCompositionInstanceRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
abstractContextComponent	<i>SwComponentPrototype</i>	0..1	ref	<b>Stereotypes:</b> atpAbstract <b>Tags:</b> xml.sequenceOffset=20
base	<i>CompositionSwComponentType</i>	0..1	ref	<b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
targetPort	<i>PortPrototype</i>	0..1	ref	<b>Stereotypes:</b> atpAbstract <b>Tags:</b> xml.sequenceOffset=30

**Table D.14: PortInCompositionTypeInstanceRef**

<b>Class</b>	<i>PPortInCompositionInstanceRef</i>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
<b>Note</b>				
<b>Base</b>	<i>ARObject</i> , <i>AtpInstanceRef</i> , <i>PortInCompositionTypeInstanceRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
contextComponent	<i>SwComponentPrototype</i>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=20
targetPPort	<i>AbstractProvidedPortPrototype</i>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30

**Table D.15: PPortInCompositionInstanceRef**

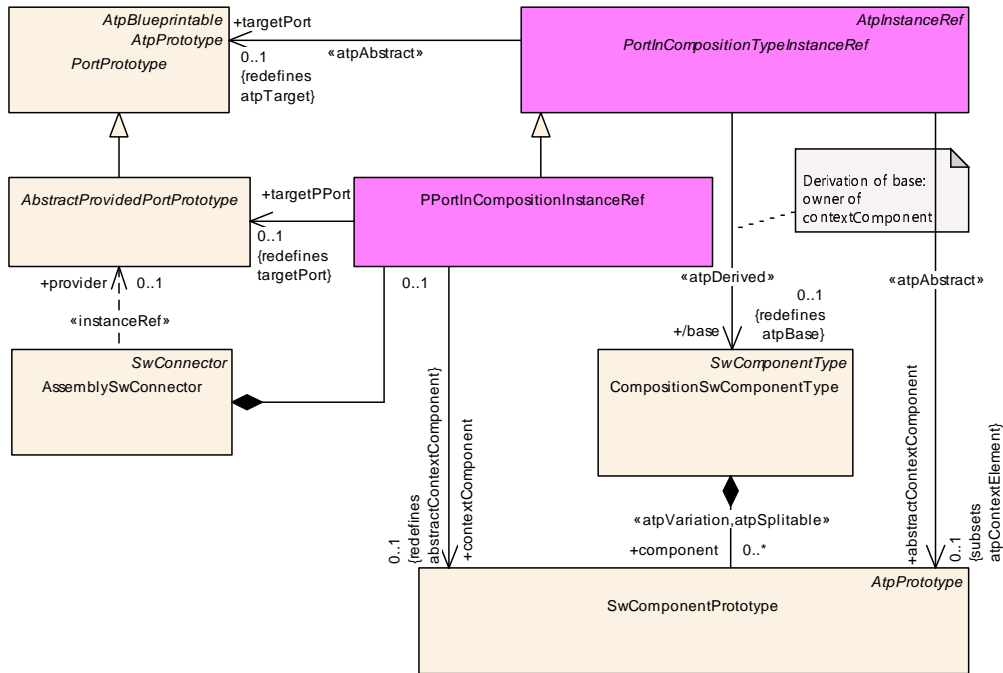


Figure D.15: Concrete modeling of references to PPortPrototype in the context of a CompositionSwComponentType

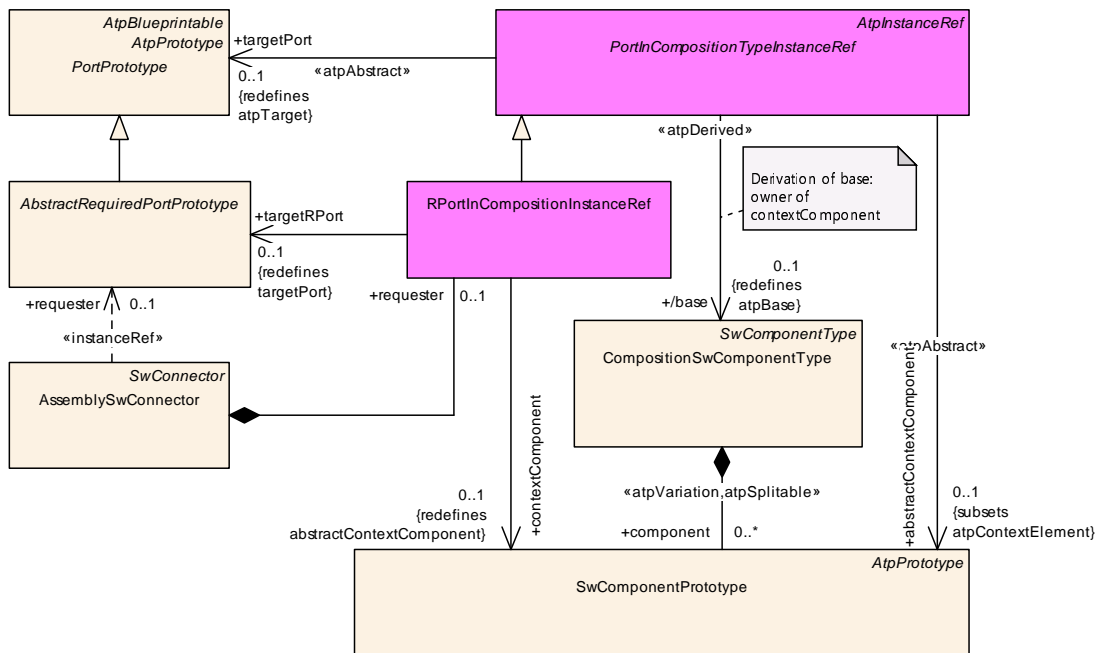


Figure D.16: Concrete modeling of references to RPortPrototype in the context of a CompositionSwComponentType

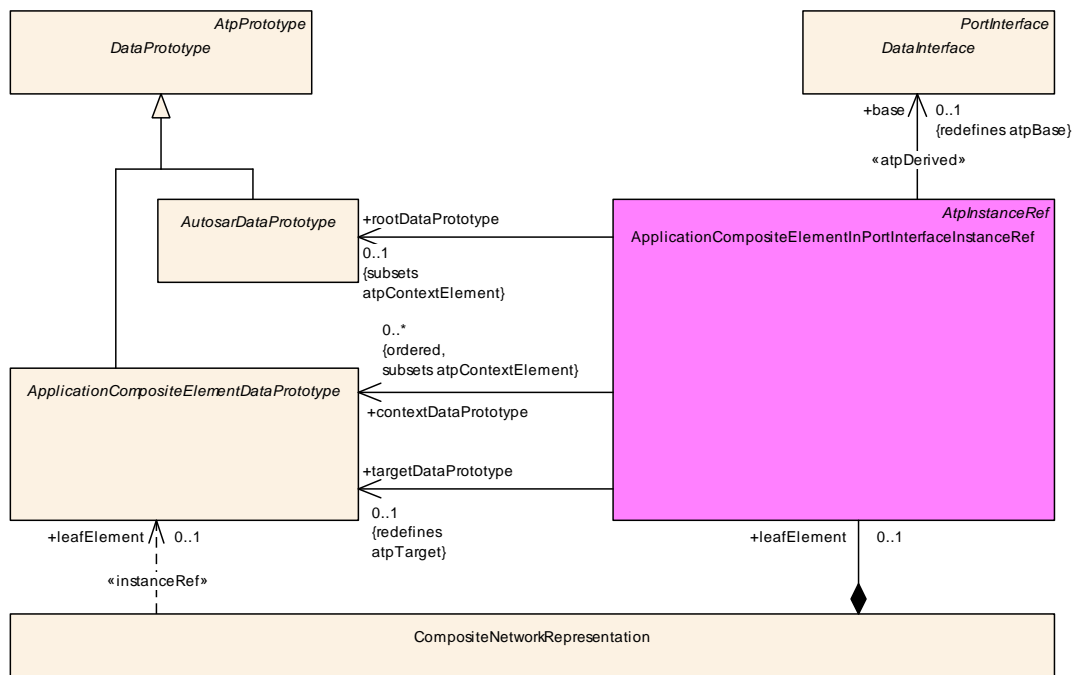
<b>Class</b>	RPortInCompositionInstanceRef
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs





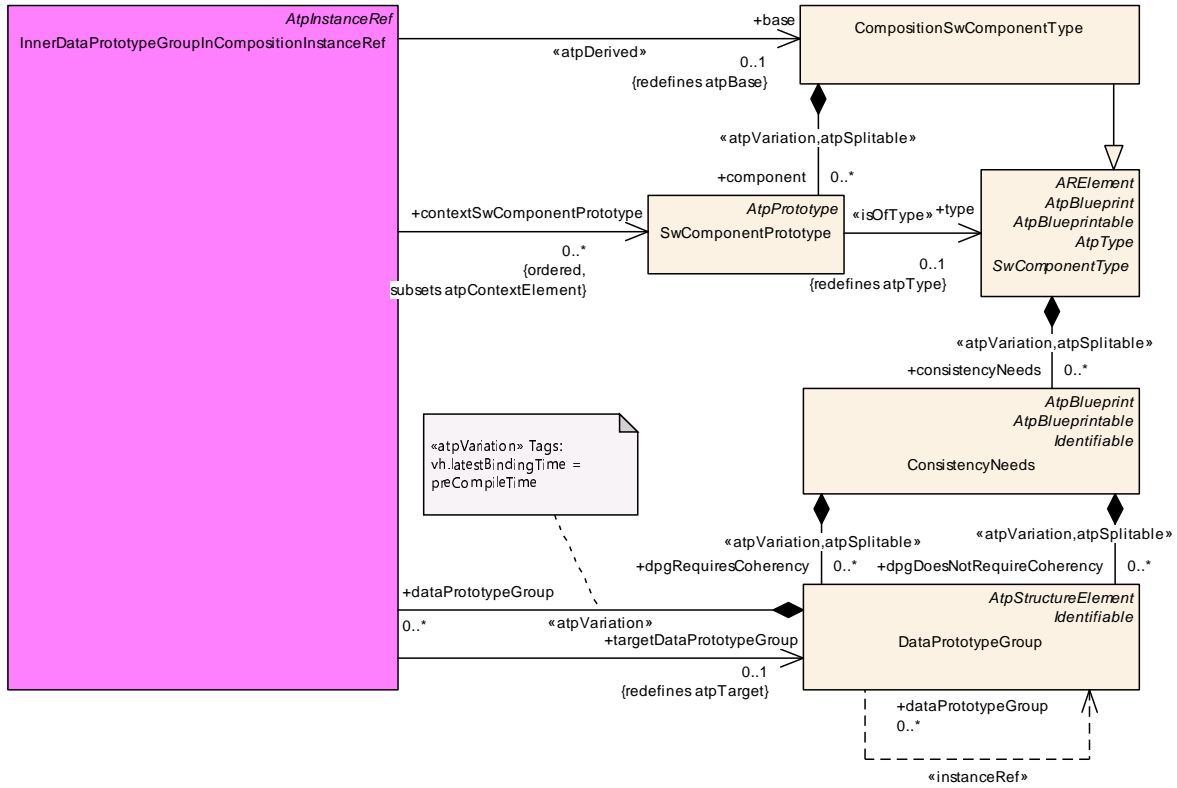
<b>Class</b>	<b>RPortInCompositionInstanceRef</b>			
<b>Note</b>				
<b>Base</b>	<i>ARObject</i> , <i>AtpInstanceRef</i> , <i>PortInCompositionTypeInstanceRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
context Component	<a href="#">SwComponent Prototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=20
targetRPort	<a href="#">AbstractRequiredPort Prototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30

**Table D.16: RPortInCompositionInstanceRef**



**Figure D.17: Modeling of references to `ApplicationCompositeElementDataPrototype` for the purpose of defining a network representation**

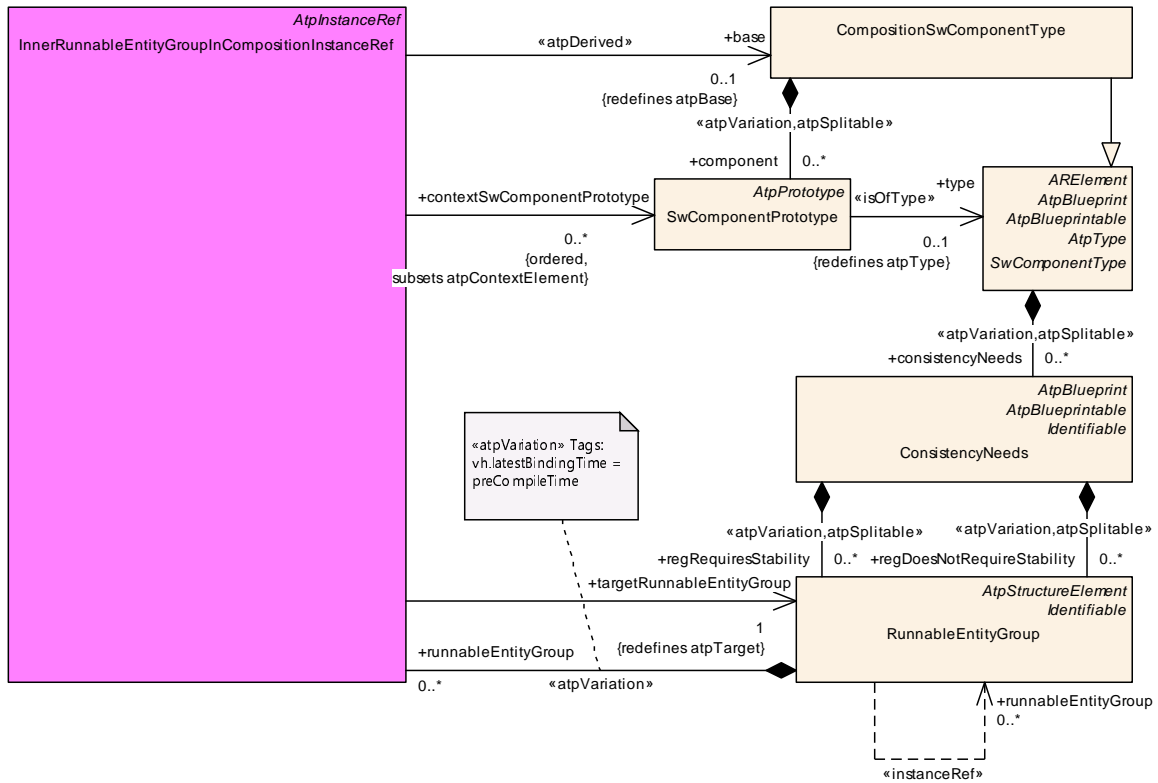
**D.2.2 Definition of implicit Communication Behavior**



**Figure D.18: Modeling of references to DataPrototypeGroup in the context of a CompositionSwComponentType**

<b>Class</b>	<b>InnerDataPrototypeGroupInCompositionInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior::InstanceRef			
<b>Note</b>	This meta-class represents the ability to define an InstanceRef to a nested DataPrototypeGroup			
<b>Base</b>	ARObject, AtpInstanceRef			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	CompositionSwComponentType	0..1	ref	This represents the base of the instanceRef. <b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
contextSwComponentPrototype (ordered)	SwComponentPrototype	*	ref	This represents the nested structure of SwComponent Prototypes. <b>Tags:</b> xml.sequenceOffset=20
targetDataPrototypeGroup	DataPrototypeGroup	0..1	ref	This represents the target of the InstanceRef <b>Tags:</b> xml.sequenceOffset=30

**Table D.17: InnerDataPrototypeGroupInCompositionInstanceRef**



**Figure D.19: Modeling of references to `RunnableEntityGroup` in the context of a `CompositionSwComponentType`**

<b>Class</b>	<code>InnerRunnableEntityGroupInCompositionInstanceRef</code>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior::InstanceRef			
<b>Note</b>	This meta-class represents the ability to define an InstanceRef to a nested RunnableEntityGroup.			
<b>Base</b>	<code>ARObject</code> , <code>AtpInstanceRef</code>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	<code>CompositionSwComponentType</code>	0..1	ref	This represents the base of the InstanceRef. <b>Stereotypes:</b> <code>atpDerived</code> <b>Tags:</b> <code>xml.sequenceOffset=10</code>
contextSwComponentPrototype (ordered)	<code>SwComponentPrototype</code>	*	ref	This represents the nested structure of SwComponent Prototypes. <b>Tags:</b> <code>xml.sequenceOffset=20</code>
targetRunnableEntityGroup	<code>RunnableEntityGroup</code>	1	ref	This represents the target association of the InstanceRef. <b>Tags:</b> <code>xml.sequenceOffset=30</code>

**Table D.18: InnerRunnableEntityGroupInCompositionInstanceRef**

<b>Class</b>	<code>RunnableEntityInCompositionInstanceRef</code>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior::InstanceRef			
<b>Note</b>	This meta-class represents the ability to define an InstanceRef to a RunnableEntity in the context of a CompositionSwComponentType.			
<b>Base</b>	<code>ARObject</code> , <code>AtpInstanceRef</code>			

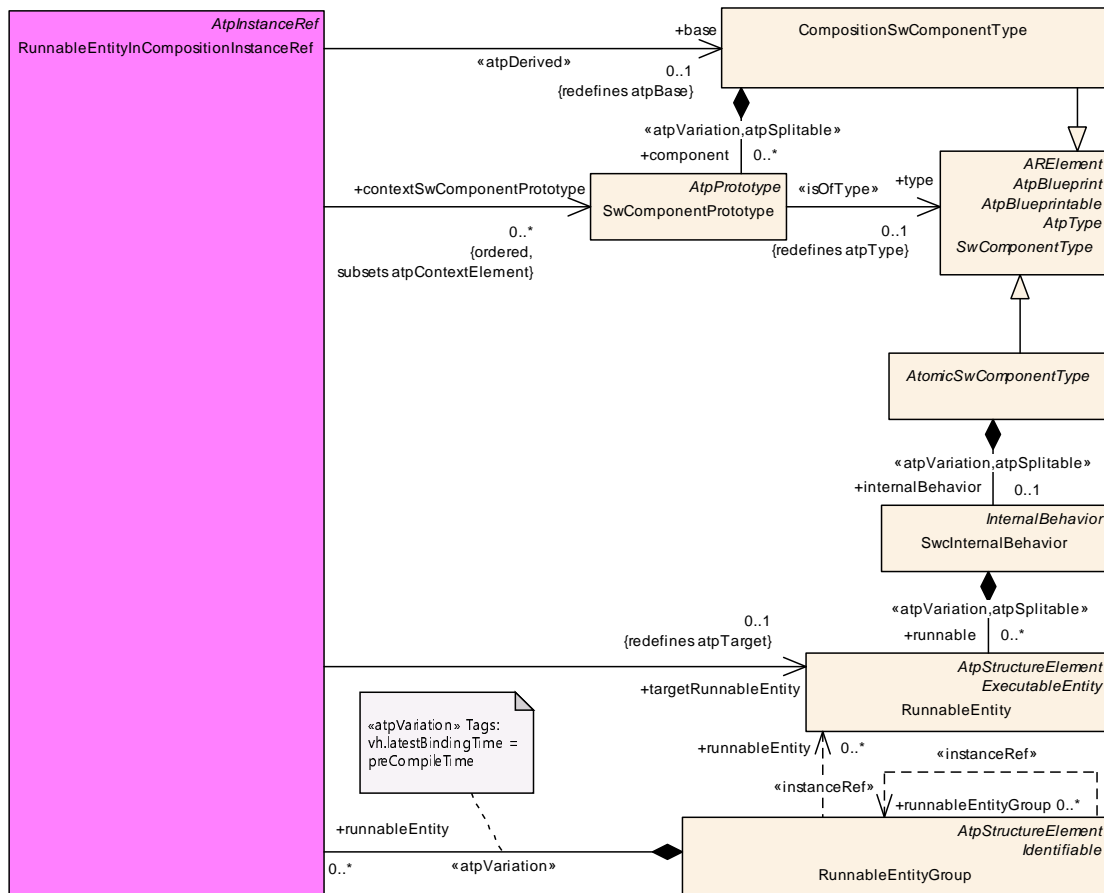




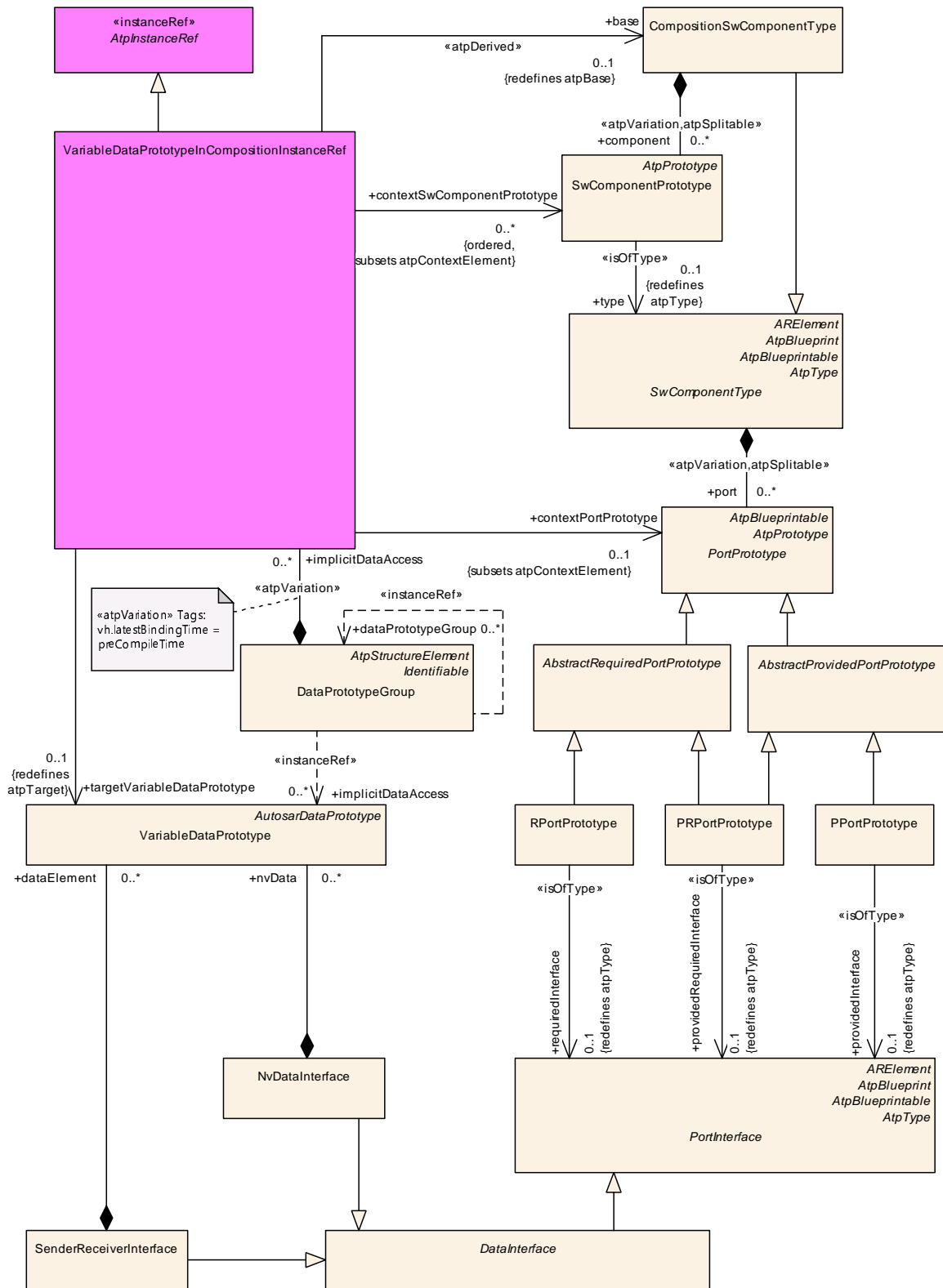


Class	RunnableEntityInCompositionInstanceRef			
Attribute	Type	Mult.	Kind	Note
base	<a href="#">CompositionSwComponentType</a>	0..1	ref	This represents the base of the InstanceRef. <b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
contextSwComponentPrototype (ordered)	<a href="#">SwComponentPrototype</a>	*	ref	This represents the nested structure of SwComponent Prototypes. <b>Tags:</b> xml.sequenceOffset=20
targetRunnableEntity	<a href="#">RunnableEntity</a>	0..1	ref	This represents the target RunnableEntity. <b>Tags:</b> xml.sequenceOffset=30

**Table D.19: RunnableEntityInCompositionInstanceRef**



**Figure D.20: Modeling of references to `RunnableEntity` in the context of a `CompositionSwComponentType` from the point of view of a `RunnableEntityGroup`**



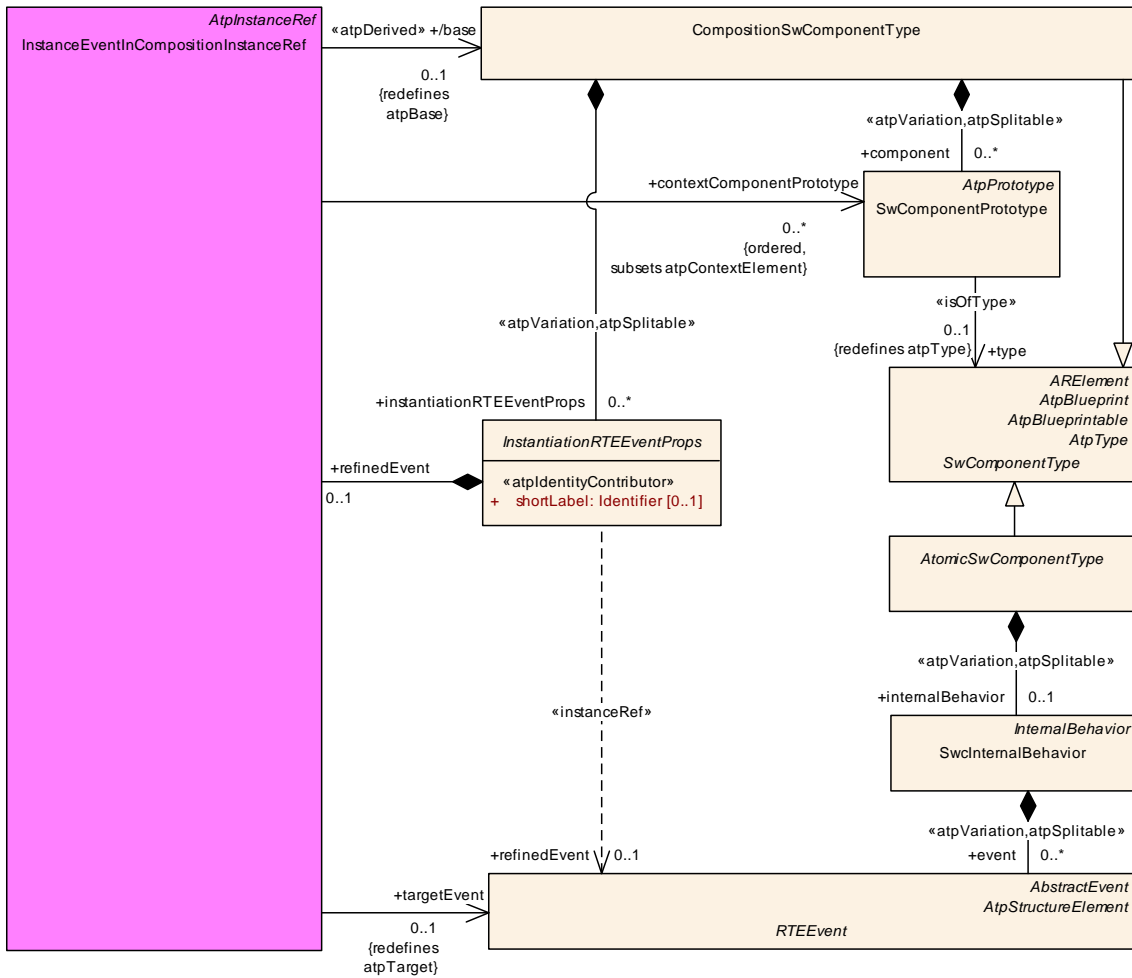
**Figure D.21: Modeling of references to *VariableDataPrototype* in the context of a *CompositionSwComponentType* from the point of view of a *RunnableEntityGroup***

<b>Class</b>	<b>VariableDataPrototypeInCompositionInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior::InstanceRef			
<b>Note</b>	This meta-class represents the ability to define an InstanceRef to a VariableDataPrototype in the context of a CompositionSwComponentType.			
<b>Base</b>	<i>ARObject</i> , <i>AtpInstanceRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	<a href="#">CompositionSwComponentType</a>	0..1	ref	This represents the base of the InstanceRef. <b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
contextPortPrototype	<a href="#">PortPrototype</a>	0..1	ref	This represents a reference to a context PortPrototype. <b>Tags:</b> xml.sequenceOffset=30
contextSwComponentPrototype (ordered)	<a href="#">SwComponentPrototype</a>	*	ref	This represents the nested structure of SwComponent Prototypes. <b>Tags:</b> xml.sequenceOffset=20
targetVariableDataPrototype	<a href="#">VariableDataPrototype</a>	0..1	ref	This represents the target VariableDataPrototype. <b>Tags:</b> xml.sequenceOffset=40

**Table D.20: VariableDataPrototypeInCompositionInstanceRef**

<b>Class</b>	<b>InstanceEventInCompositionInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
<b>Note</b>				
<b>Base</b>	<i>ARObject</i> , <i>AtpInstanceRef</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	<a href="#">CompositionSwComponentType</a>	0..1	ref	<b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
contextComponentPrototype (ordered)	<a href="#">SwComponentPrototype</a>	*	ref	<b>Tags:</b> xml.sequenceOffset=20
targetEvent	<a href="#">RTEEvent</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30

**Table D.21: InstanceEventInCompositionInstanceRef**



**Figure D.22: Modeling of references to `RTEEvent` in the context of a `InstantiationRTEEventProps` from the point of view of a `CompositionSwComponentType`**

### D.2.3 Internal Behavior

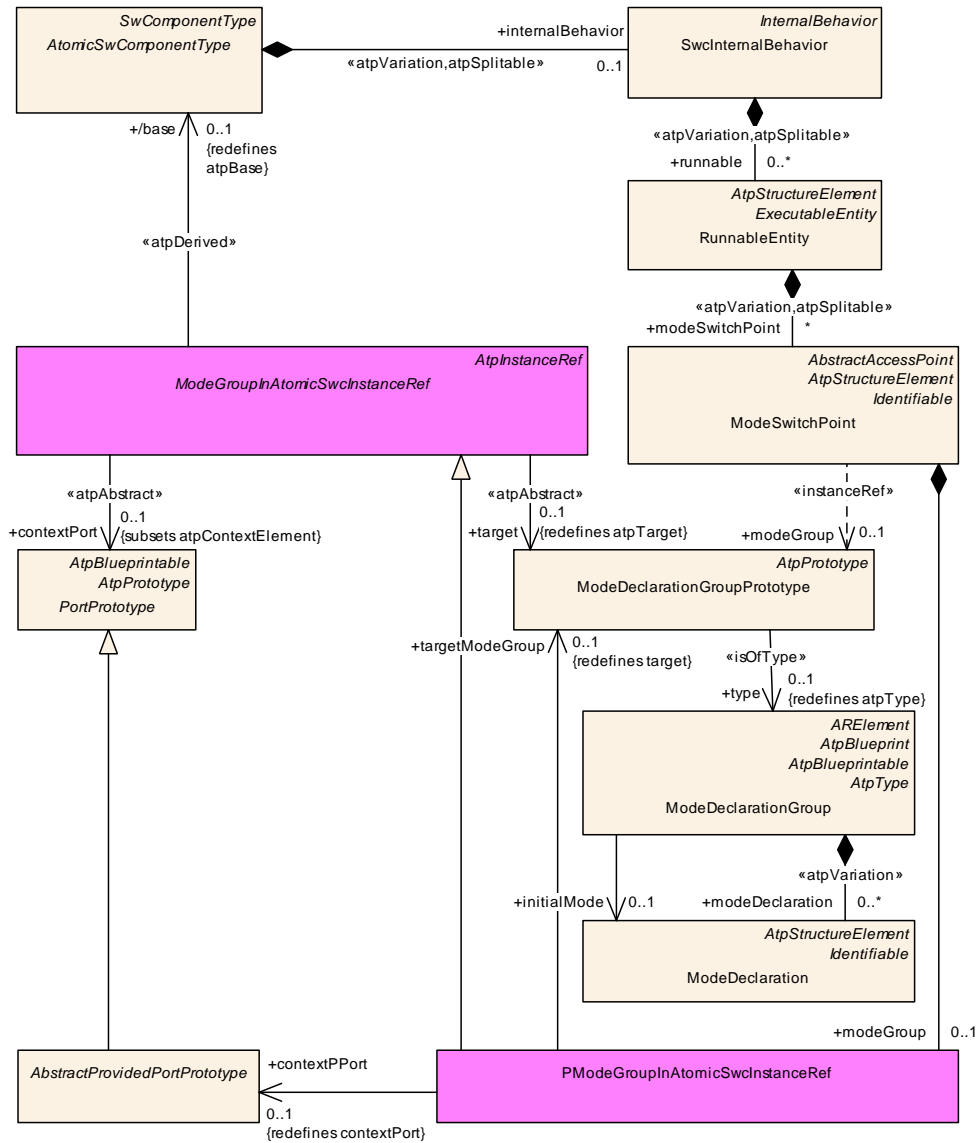


Figure D.23: Modeling of references to provided **ModeDeclarationGroupPrototype** in the context of an **AtomicSwComponentType**

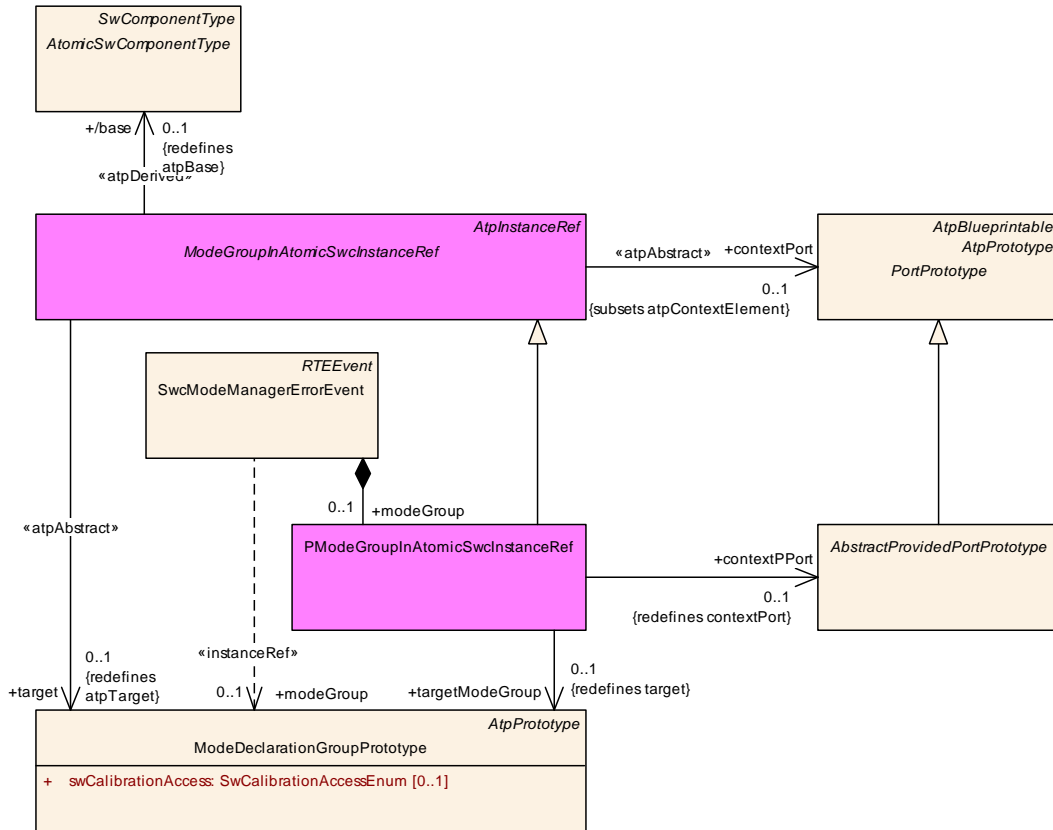
<b>Class</b>	<b>ModeGroupInAtomicSwInstanceRef</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
<b>Note</b>				
<b>Base</b>	ARObject, AtpInstanceRef			
<b>Subclasses</b>	PModeGroupInAtomicSwInstanceRef, RModeGroupInAtomicSWCInstanceRef			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	AtomicSwComponentType	0..1	ref	<b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
contextPort	PortPrototype	0..1	ref	<b>Stereotypes:</b> atpAbstract <b>Tags:</b> xml.sequenceOffset=20





<b>Class</b>	<b>ModeGroupInAtomicSwcInstanceRef</b> (abstract)			
target	<a href="#">ModeDeclarationGroup Prototype</a>	0..1	ref	<b>Stereotypes:</b> atpAbstract <b>Tags:</b> xml.sequenceOffset=30

**Table D.22: ModeGroupInAtomicSwcInstanceRef**



**Figure D.24: Modeling of references to provided ModeDeclarationGroupPrototype to be used by SwcModeManagerErrorEvent**



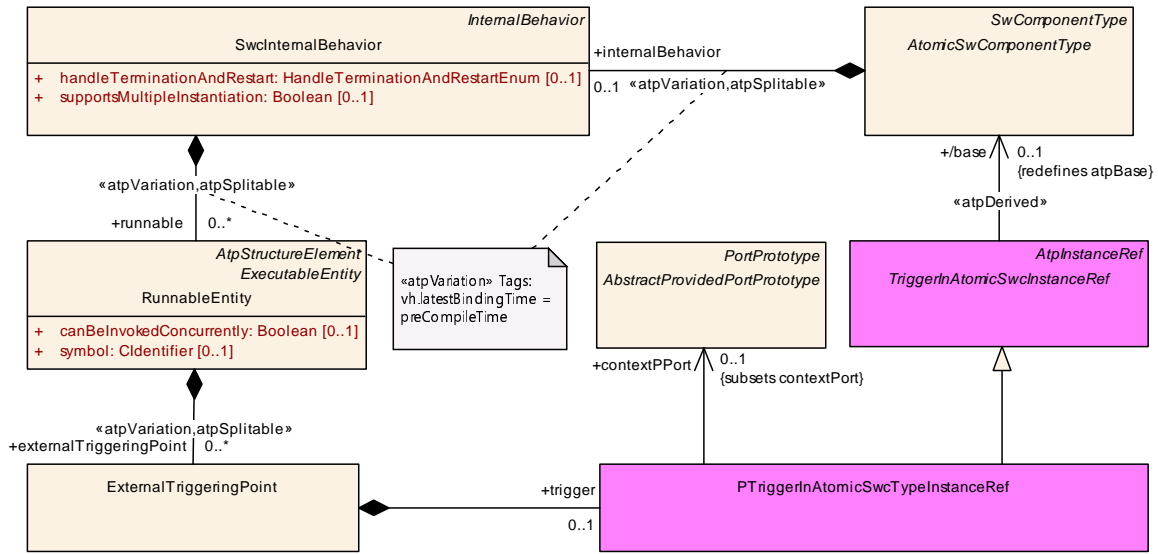


Figure D.27: Modeling of references to a **Trigger** in the context of a **SwComponentType**

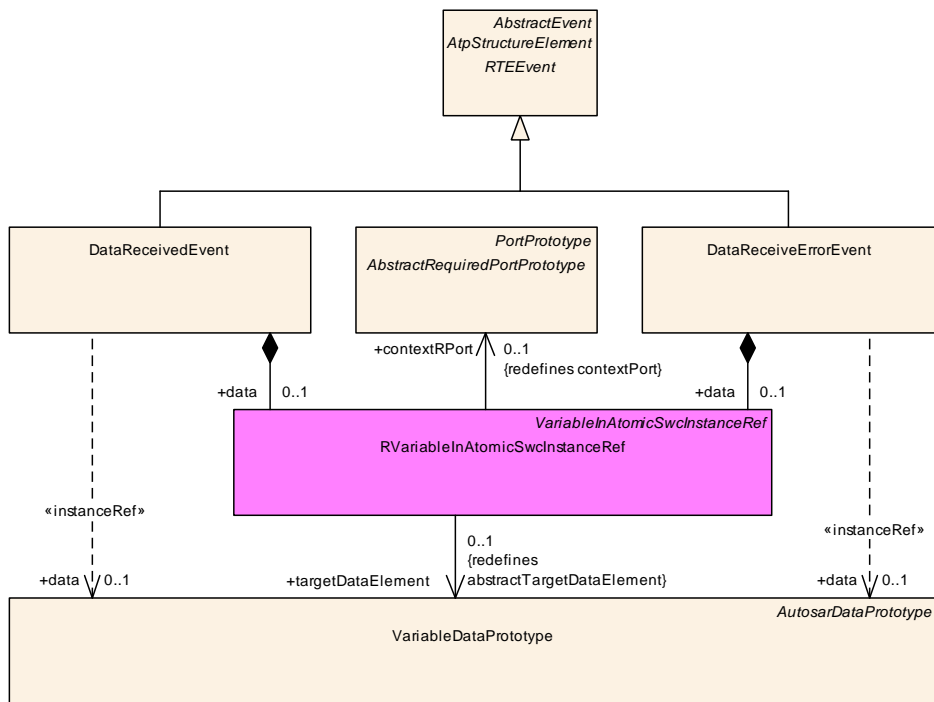
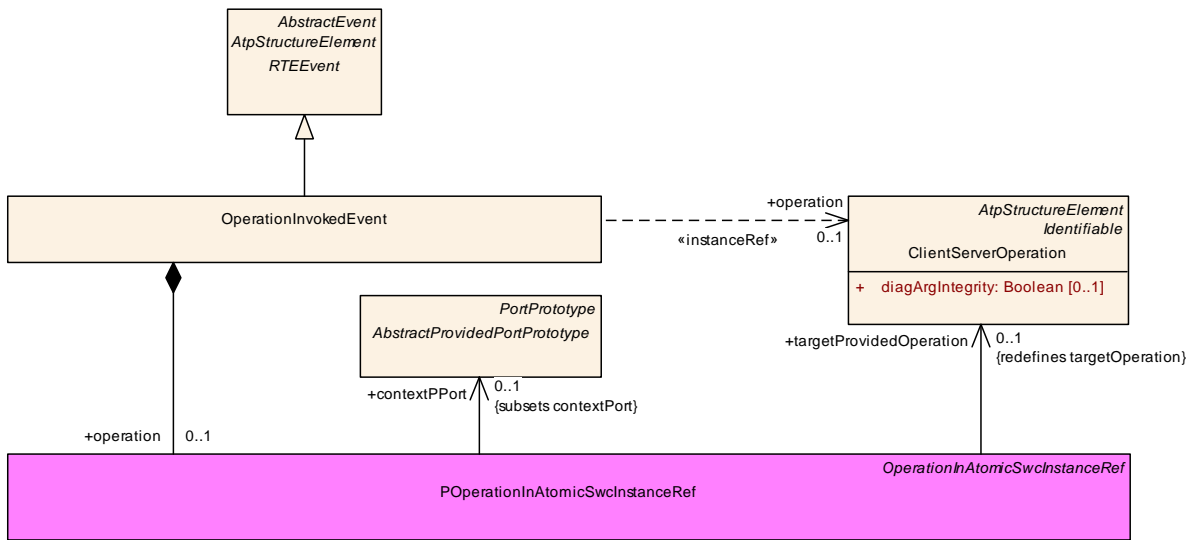
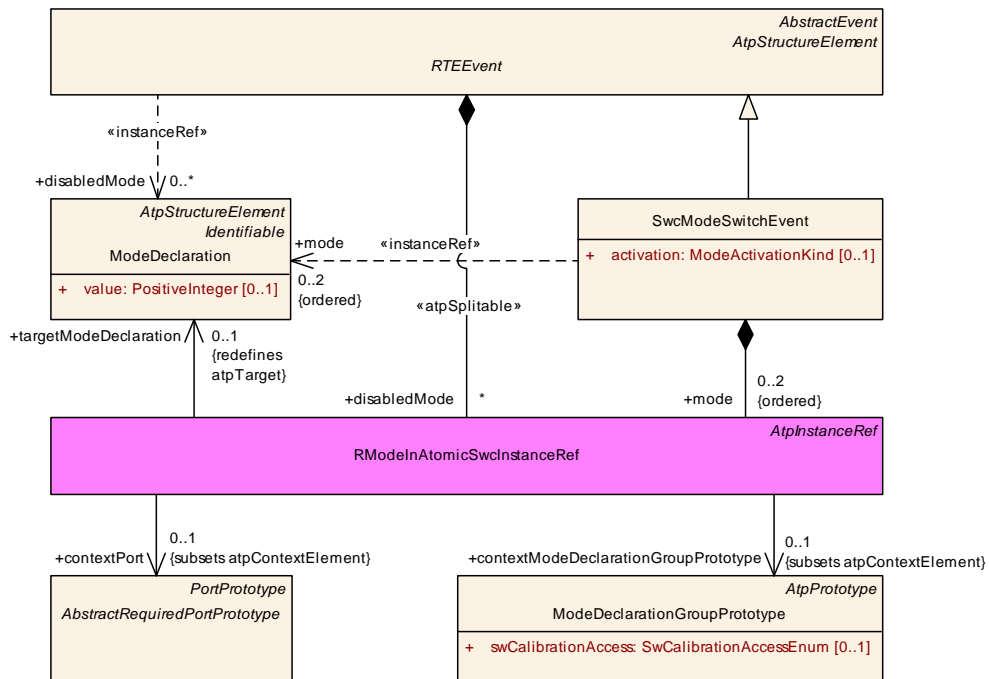


Figure D.28: Modeling of references to a **VariableDataPrototype** used in the context of an **RTEEvent**

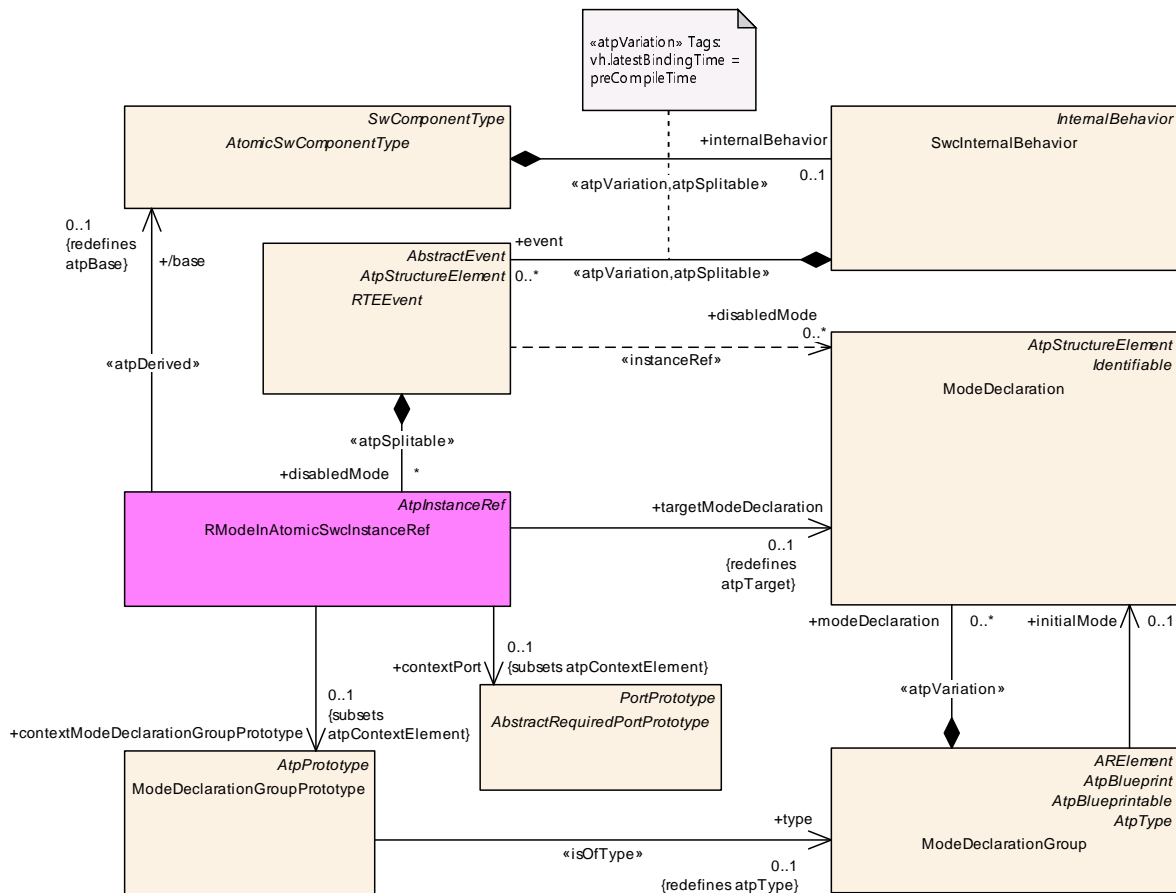




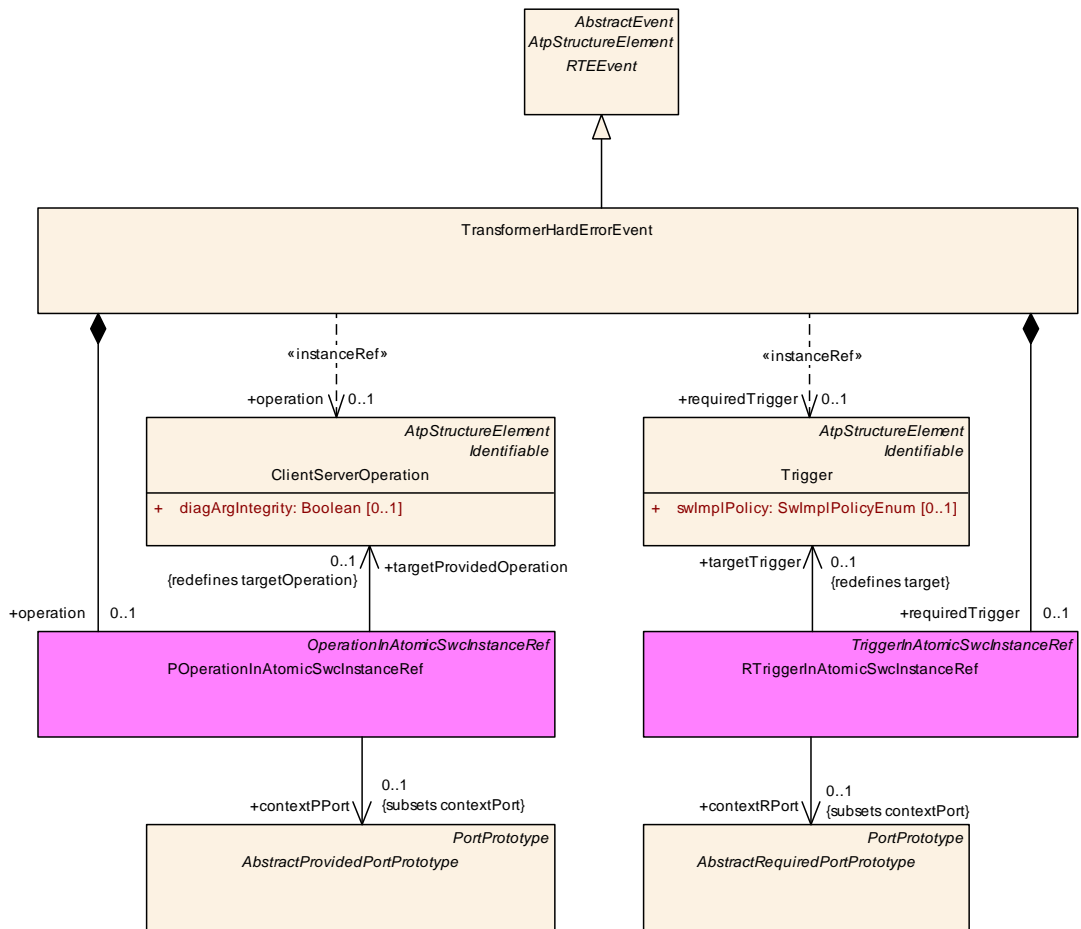
**Figure D.29: Modeling of references to a *ClientServerOperation* used in the context of an *RTEEvent***



**Figure D.30: Modeling of references to a *ModeDeclaration* used in the context of an *RTEEvent***



**Figure D.31: Modeling of mode disabling**



**Figure D.32: 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](#).

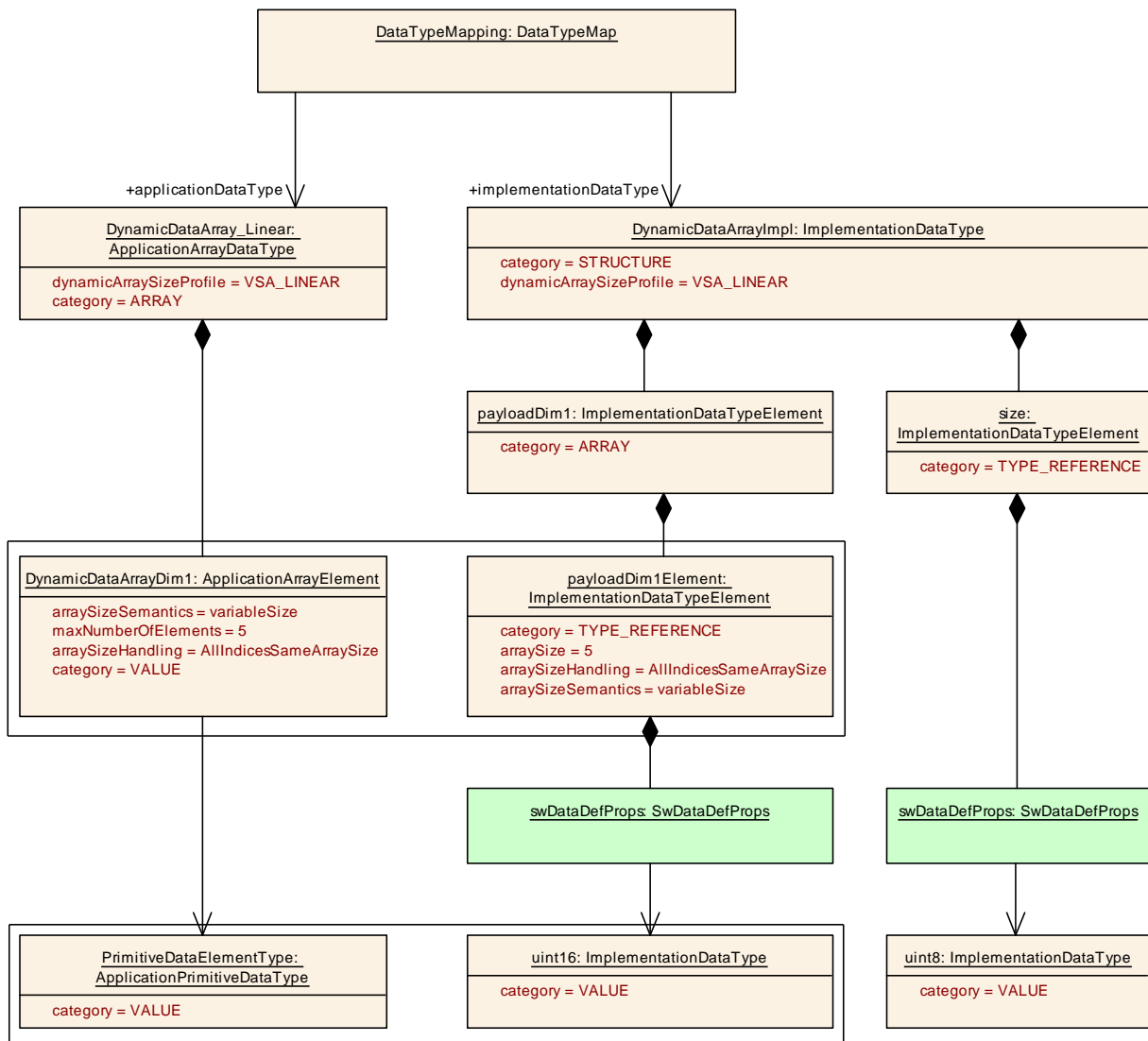
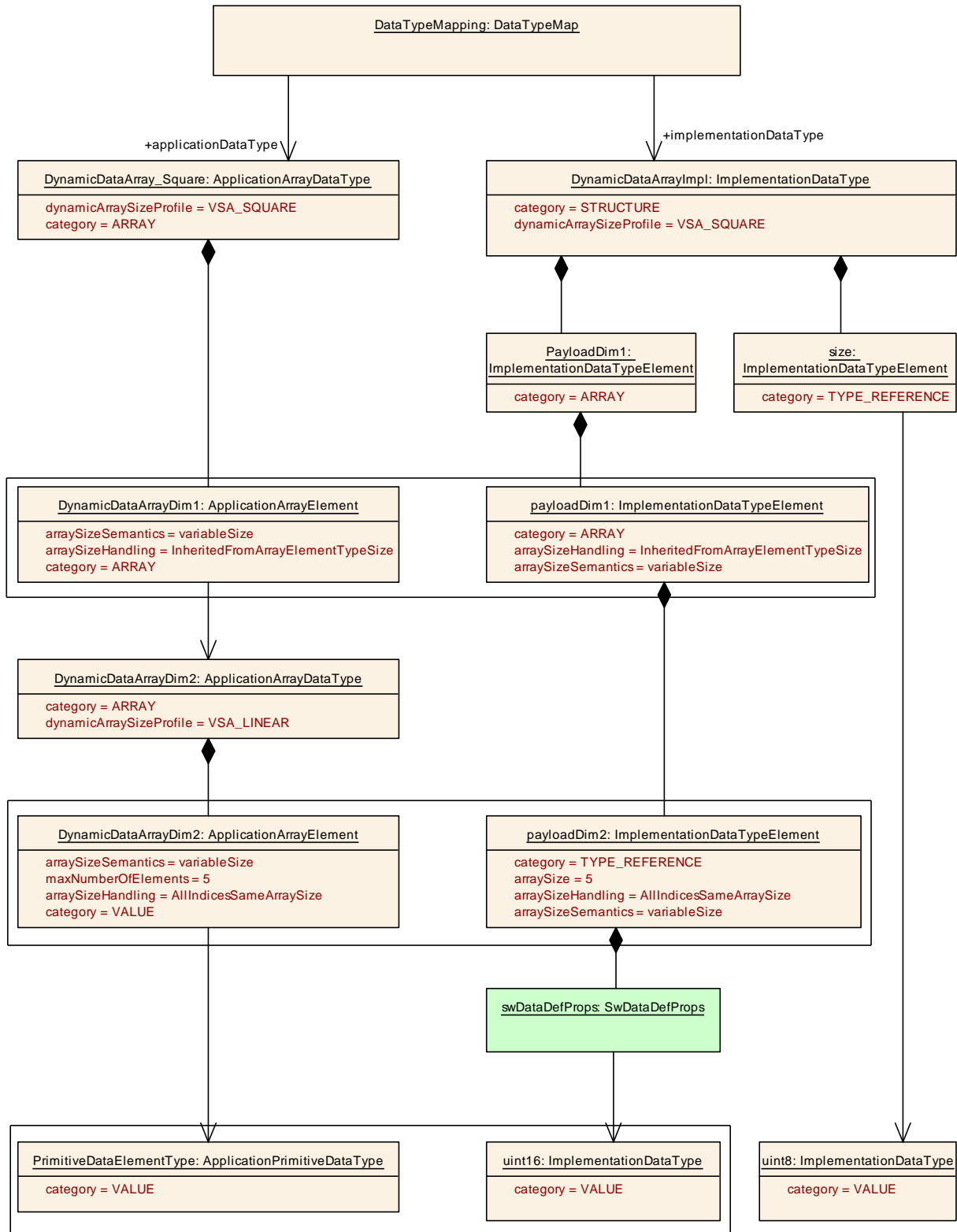
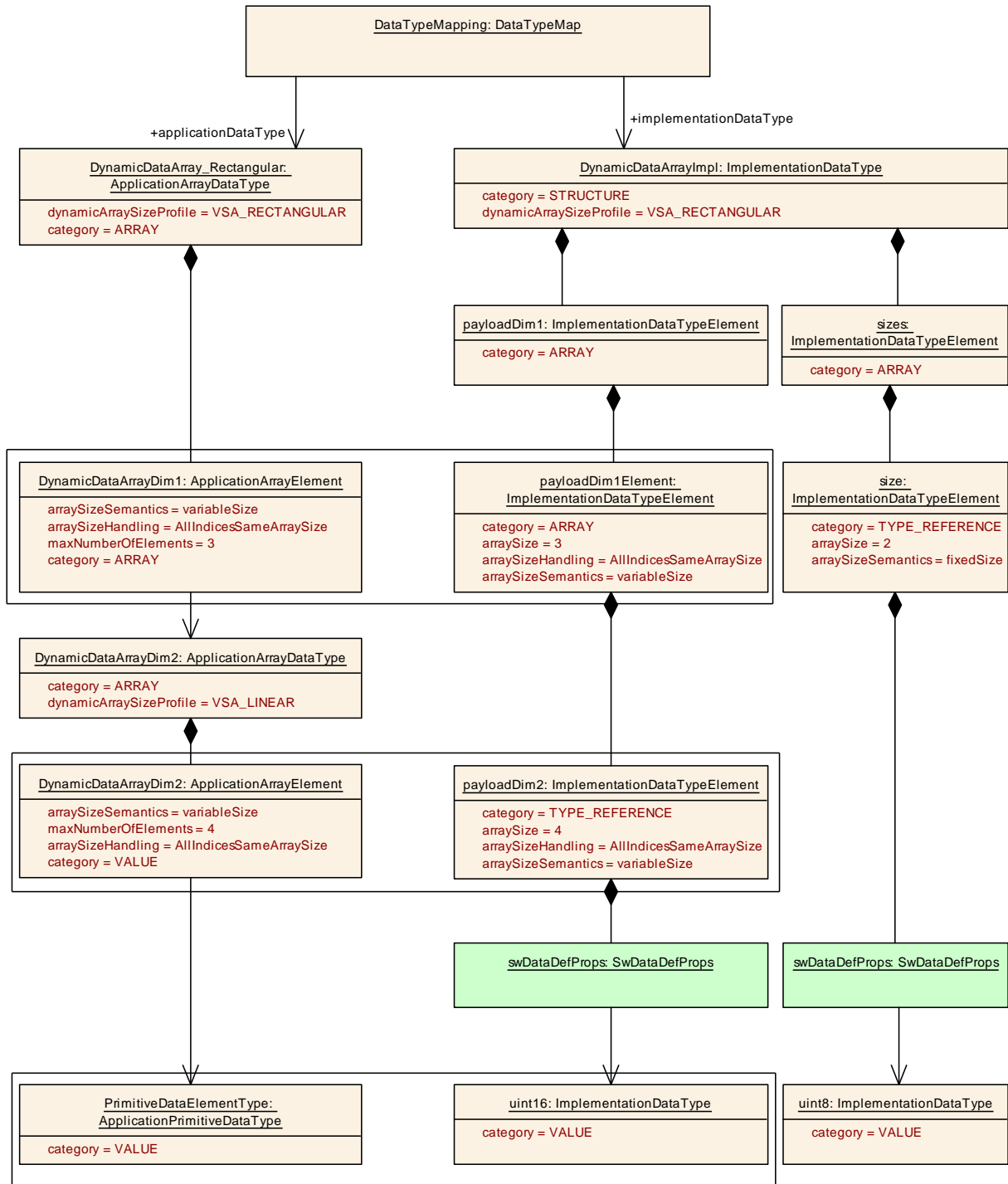


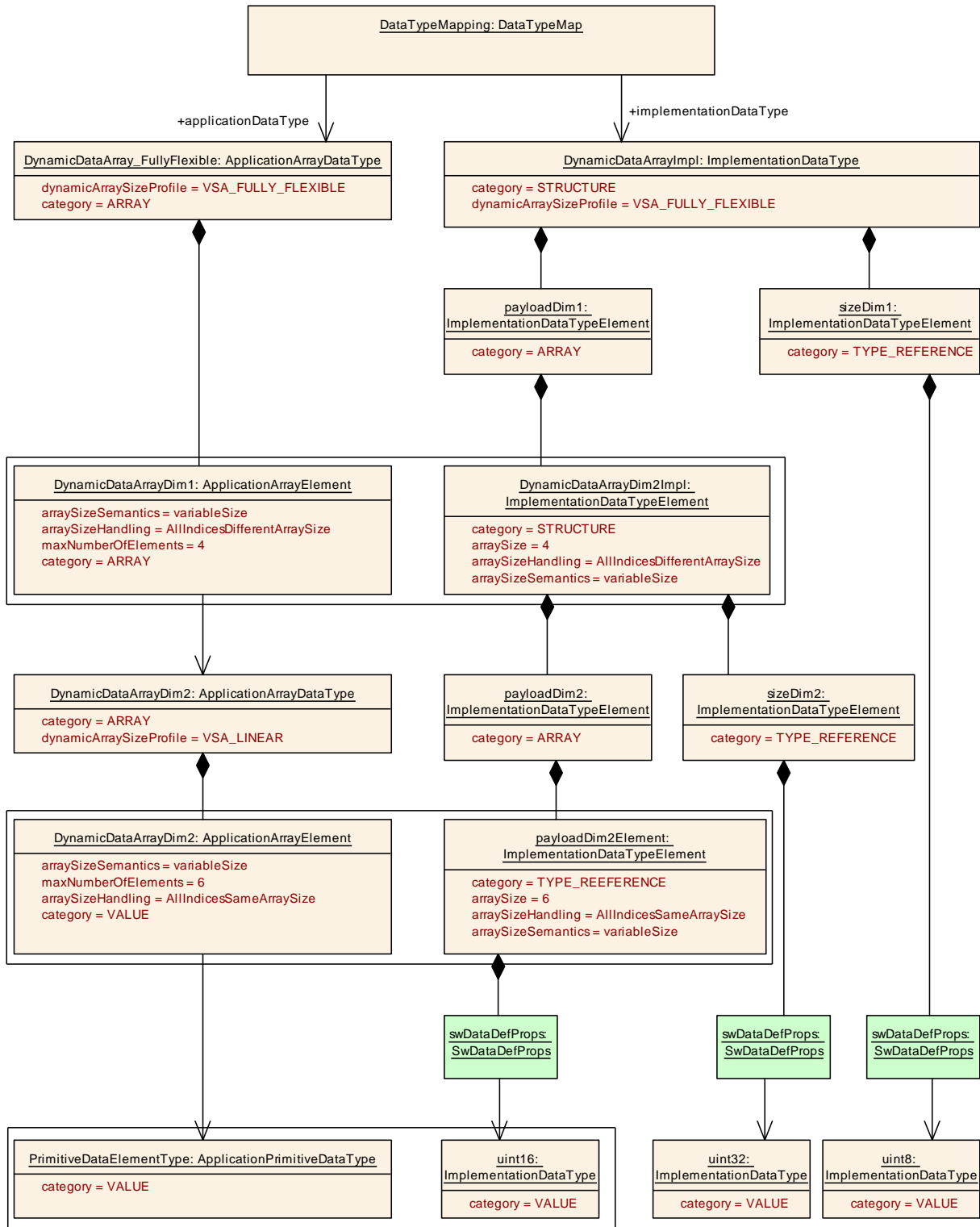
Figure E.1: Example of a linear variable size array



**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.

<b>Class</b>	<b>ARElement</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
<b>Note</b>	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
<b>Base</b>	<i>ARObject</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Subclasses</b>	AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, ApplicationPartition, <a href="#">AutosarDataType</a> , <a href="#">BaseType</a> , BlueprintMappingSet, BswEntryRelationshipSet, <a href="#">BswModuleDescription</a> , <a href="#">BswModuleEntry</a> , BuildActionManifest, <a href="#">CalibrationParameterValueSet</a> , ClientIdDefinitionSet, ClientServerInterfaceToBswModuleEntryBlueprintMapping, Collection, <a href="#">CompuMethod</a> , ConsistencyNeedsBlueprintSet, <a href="#">ConstantSpecification</a> , <a href="#">ConstantSpecificationMappingSet</a> , CpSoftwareCluster, CpSoftwareClusterBinaryManifestDescriptor, CpSoftwareClusterMappingSet, CpSoftwareClusterResourcePool, CryptoEllipticCurveProps, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, CryptoSignatureScheme, <a href="#">DataConstr</a> , DataExchangePoint, DataTransformationSet, <a href="#">DataTypeMappingSet</a> , <a href="#">DiagnosticCommonElement</a> , DiagnosticConnection, DiagnosticContributionSet, DltContext, DltEcu, Documentation, <a href="#">E2EProfileCompatibilityProps</a> , EcuDefinitionCollection, EcuDestinationUriDefSet, EcuModuleConfigurationValues, EcuModuleDef, <a href="#">EcuValueCollection</a> , <a href="#">EndToEndProtectionSet</a> , EthIhpProps, EthTcplpIcmpProps, EthTcplpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FlatMap, GeneralPurposeConnection, HwCategory, <a href="#">HwElement</a> , <a href="#">HwType</a> , IPsecConfigProps, IPv6ExtHeaderFilterSet, <i>IdsCommonElement</i> , IdsDesign, <a href="#">Implementation</a> , <a href="#">InterpolationRoutineMappingSet</a> , J1939ControllerApplication, KeywordSet, LifecycleInfoSet, LifecycleStateDefinitionGroup, LogAndTraceMessageCollectionSet, McFunction, McGroup, <a href="#">ModeDeclarationGroup</a> , <a href="#">ModeDeclarationMappingSet</a> , OsTaskProxy, <a href="#">PhysicalDimension</a> , <a href="#">PhysicalDimensionMappingSet</a> , <a href="#">PortInterface</a> , <a href="#">PortInterfaceMappingSet</a> , PortPrototypeBlueprint, <a href="#">PostBuildVariantCriterion</a> , <a href="#">PostBuildVariantCriterionValueSet</a> , PredefinedVariant, <a href="#">RapidPrototypingScenario</a> , SdgDef, SignalServiceTranslationPropsSet, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, <a href="#">SwAddrMethod</a> , <a href="#">SwAxisType</a> , SwComponentMappingConstraints, <a href="#">SwComponentType</a> , <a href="#">SwRecordLayout</a> , <a href="#">SwSystemconst</a> , <a href="#">SwSystemconstantValueSet</a> , <a href="#">SwCswMapping</a> , <a href="#">System</a> , <a href="#">SystemSignal</a> , SystemSignalGroup, TDCpSoftwareClusterMappingSet, TcpOptionFilterSet, <a href="#">TimingExtension</a> , TlsConnectionGroup, TlvDataIdDefinitionSet, TransformationPropsSet, <a href="#">Unit</a> , <a href="#">UnitGroup</a> , ViewMapSet			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table F.1: ARElement**

<b>Class</b>	<b>ARPackage</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>







Class	ARPackage			
arPackage	<a href="#">ARPackage</a>	*	aggr	<p>This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=arPackage.shortName, arPackage.variationPoint.shortLabel            vh.latestBindingTime=blueprintDerivationTime            xml.sequenceOffset=30</p>
element	PackageableElement	*	aggr	<p>Elements that are part of this package</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=element.shortName, element.definition, element.variationPoint.shortLabel            vh.latestBindingTime=systemDesignTime            xml.sequenceOffset=20</p>
referenceBase	ReferenceBase	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>            atp.Splitkey=referenceBase.shortLabel            xml.sequenceOffset=10</p>

**Table F.2: ARPackage**

Class	AUTOSAR			
Package	M2::AUTOSARTemplates::AutosarTopLevelStructure			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. <b>Tags:</b> xml.globalElement=true			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
adminData	<a href="#">AdminData</a>	0..1	aggr	<p>This represents the administrative data of an Autosar file.</p> <p><b>Tags:</b>xml.sequenceOffset=10</p>
arPackage	<a href="#">ARPackage</a>	*	aggr	<p>This is the top level package in an AUTOSAR model.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>            atp.Splitkey=arPackage.shortName, arPackage.variationPoint.shortLabel            vh.latestBindingTime=blueprintDerivationTime            xml.sequenceOffset=30</p>
fileInfo Comment	FileInfoComment	0..1	aggr	<p>This represents a possibility to provide a structured comment in an AUTOSAR file.</p> <p><b>Stereotypes:</b> atpStructuredComment  <b>Tags:</b>            xml.roleElement=true            xml.sequenceOffset=-10            xml.typeElement=false</p>
introduction	<a href="#">DocumentationBlock</a>	0..1	aggr	<p>This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>

**Table F.3: AUTOSAR**

<b>Class</b>	<<atpMixedString>> <b>AbstractNumericalVariationPoint</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling::AttributeValueVariationPoints			
<b>Note</b>	This is an abstract NumericalValueVariationPoint. It is introduced to support the case that additional attributes are required for particular purposes.			
<b>Base</b>	ARObject, <a href="#">AttributeValueVariationPoint</a> , <a href="#">FormulaExpression</a> , <a href="#">SwSystemconstDependentFormula</a>			
<b>Subclasses</b>	LimitValueVariationPoint, NumericalValueVariationPoint			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table F.4: AbstractNumericalVariationPoint**

<b>Class</b>	<b>AdminData</b>			
<b>Package</b>	M2::MSR::AsamHdo::AdminData			
<b>Note</b>	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 <ul style="list-style-type: none"> <li>• The language and/or used languages.</li> <li>• 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.</li> <li>• Document meta-data specific for a company</li> </ul>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=50 xml.typeElement=false xml.typeWrapperElement=false
language	LEnum	0..1	attr	This attribute specifies the master language of the document or the document fragment. The master language is the one in which the document is maintained and from which the other languages are derived from. In particular in case of inconsistencies, the information in the master language is priority.  <b>Tags:</b> xml.sequenceOffset=20
sdg	<a href="#">Sdg</a>	*	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.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=sdg, sdg.variationPoint.shortLabel xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=60 xml.typeElement=false xml.typeWrapperElement=false





Class	AdminData			
usedLanguages	MultiLanguagePlainText	0..1	aggr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>

**Table F.5: 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	Type	Mult.	Kind	Note
base	AtpClassifier	1	ref	<p>This is the base from which navigation path begins.</p> <p><b>Stereotypes:</b> atpDerived</p>
contextElement (ordered)	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.6: AnyInstanceRef**

Class	ApplicationCompositeElementInPortInterfaceInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef			
Attribute	Type	Mult.	Kind	Note
base	DataInterface	0..1	ref	<p>This represents the SenderReceiverInterface that acts as the base in this InstanceRef definition</p> <p><b>Stereotypes:</b> atpDerived</p> <p><b>Tags:</b>xml.sequenceOffset=10</p>
contextData Prototype (ordered)	ApplicationCompositeElementDataPrototype	*	ref	<p>This represents a context ApplicationCompositeData Prototype</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>
rootData Prototype	AutosarDataPrototype	0..1	ref	<p>This refers to the dataPrototype which is typed by the ApplicationDatatype in which which the target can be found.</p> <p><b>Tags:</b>xml.sequenceOffset=15</p>
targetData Prototype	ApplicationCompositeElementDataPrototype	0..1	ref	<p>This represents the referenced ApplicationComposite DataPrototype.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>

**Table F.7: ApplicationCompositeElementInPortInterfaceInstanceRef**

<b>Class</b>	<b>AtpInstanceRef</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::AbstractStructure			
<b>Note</b>	<p>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.</p> <p>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).</p>			
<b>Base</b>	AObject			
<b>Subclasses</b>	<a href="#">AnyInstanceRef</a> , <a href="#">ApplicationCompositeElementInPortInterfaceInstanceRef</a> , <a href="#">ComponentInCompositionInstanceRef</a> , <a href="#">ComponentInSystemInstanceRef</a> , <a href="#">DataPrototypeInPortInterfaceInstanceRef</a> , <a href="#">DataPrototypeInSystemInstanceRef</a> , <a href="#">InnerDataPrototypeGroupInCompositionInstanceRef</a> , <a href="#">InnerPortGroupInCompositionInstanceRef</a> , <a href="#">InnerRunnableEntityGroupInCompositionInstanceRef</a> , <a href="#">InstanceEventInCompositionInstanceRef</a> , <a href="#">ModeDeclarationGroupPrototypeInSystemInstanceRef</a> , <a href="#">ModeGroupInAtomicSwcInstanceRef</a> , <a href="#">ModelInBswModuleDescriptionInstanceRef</a> , <a href="#">ModelInSwcInstanceRef</a> , <a href="#">OperationArgumentInComponentInstanceRef</a> , <a href="#">OperationInAtomicSwcInstanceRef</a> , <a href="#">OperationInSystemInstanceRef</a> , <a href="#">PModelInSystemInstanceRef</a> , <a href="#">ParameterDataPrototypeInSystemInstanceRef</a> , <a href="#">ParameterInAtomicSWCTypeInstanceRef</a> , <a href="#">PortGroupInSystemInstanceRef</a> , <a href="#">PortInCompositionTypeInstanceRef</a> , <a href="#">RModelInAtomicSwcInstanceRef</a> , <a href="#">RteEventInCompositionInstanceRef</a> , <a href="#">RteEventInEcuInstanceRef</a> , <a href="#">RteEventInSystemInstanceRef</a> , <a href="#">RunnableEntityInCompositionInstanceRef</a> , <a href="#">SwcServiceDependencyInSystemInstanceRef</a> , <a href="#">TriggerInAtomicSwcInstanceRef</a> , <a href="#">TriggerInSystemInstanceRef</a> , <a href="#">VariableAccessInEcuInstanceRef</a> , <a href="#">VariableDataPrototypeInCompositionInstanceRef</a> , <a href="#">VariableDataPrototypeInSystemInstanceRef</a> , <a href="#">VariableInAtomicSWCTypeInstanceRef</a> , <a href="#">VariableInAtomicSwcInstanceRef</a> , <a href="#">VariableInComponentInstanceRef</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
atpBase	AtpClassifier	1	ref	<p>This is the base from which the navigation path starts.</p> <p><b>Stereotypes:</b> atpAbstract; atpDerived</p>
atpContext Element (ordered)	AtpPrototype	*	ref	<p>This is one particular step in the navigation path.</p> <p><b>Stereotypes:</b> atpAbstract</p>
atpTarget	AtpFeature	1	ref	<p>This is the target of the instance ref. In other words it is the terminal of the navigation path.</p> <p><b>Stereotypes:</b> atpAbstract</p>

**Table F.8: AtpInstanceRef**

<b>Enumeration</b>	<b>BindingTimeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling
<b>Note</b>	This enumerator specifies the applicable binding times for the pre build variation points.
<b>Literal</b>	<b>Description</b>
codeGenerationTime	<ul style="list-style-type: none"> <li>Coding by hand, based on requirements document.</li> <li>Tool based code generation, e.g. from a model.</li> <li>The model may contain variants.</li> <li>Only code for the selected variant(s) is actually generated.</li> </ul> <p><b>Tags:</b>atp.EnumerationLiteralIndex=0</p>
linkTime	<p>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)</p> <p><b>Tags:</b>atp.EnumerationLiteralIndex=1</p>
preCompileTime	<p>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.</p> <p><b>Tags:</b>atp.EnumerationLiteralIndex=2</p>





Enumeration	BindingTimeEnum
systemDesignTime	<ul style="list-style-type: none"> <li>Designing the VFB.</li> <li>Software Component types (PortInterfaces).</li> <li>SWC Prototypes and the Connections between SWCprototypes.</li> <li>Designing the Topology</li> <li>ECUs and interconnecting Networks</li> <li>Designing the Communication Matrix and Data Mapping</li> </ul> <p><b>Tags:</b>atp.EnumerationLiteralIndex=3</p>

**Table F.9: BindingTimeEnum**

Class	BswCalledEntity			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
<b>Note</b>	BSW module entity which is designed to be called from another BSW module or cluster.			
<b>Base</b>	<i>ARObject, BswModuleEntity, ExecutableEntity, Identifiable, MultilanguageReferrable, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

**Table F.10: BswCalledEntity**

Class	BswImplementation			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswImplementation			
<b>Note</b>	Contains the implementation specific information in addition to the generic specification (BswModule Description and BswBehavior). It is possible to have several different BswImplementations referring to the same BswBehavior. <b>Tags:</b> atp.recommendedPackage=BswImplementations			
<b>Base</b>	<i>ARElement, ARObject, CollectableElement, Identifiable, Implementation, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	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.
behavior	<a href="#">BswInternalBehavior</a>	1	ref	The behavior of this implementation. This relation is made as an association because <ul style="list-style-type: none"> <li>it follows the pattern of the SWCT</li> <li>since ARElement cannot be splitted, but we want supply the implementation later, the Bsw Implementation is not aggregated in BswBehavior</li> </ul>
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. <b>Tags:</b> xml.roleWrapperElement=true
recommendedConfiguration	EcucModuleConfigurationValues	*	ref	Reference to one or more sets of recommended configuration values for this module or module cluster.





Class	BswImplementation			
vendorApiInfix	Identifier	0..1	attr	<p>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: &lt;Module Name&gt;_&lt;vendorId&gt;_&lt;vendorApiInfix&gt;_&lt;API name from SWS&gt;.</p> <p>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.</p> <p>This attribute is mandatory for all modules with upper multiplicity &gt; 1. It shall not be used for modules with upper multiplicity =1.</p> <p>See also SWS_BSW_00102.</p>
vendorSpecificModuleDef	EcucModuleDef	*	ref	<p>Reference to</p> <ul style="list-style-type: none"> <li>the vendor specific EcucModuleDef used in this BswImplementation if it represents a single module</li> <li>several EcucModuleDefs used in this BswImplementation if it represents a cluster of modules</li> <li>one or no EcucModuleDefs used in this BswImplementation if it represents a library</li> </ul> <p><b>Tags:</b>xml.roleWrapperElement=true</p>

**Table F.11: BswImplementation**

Class	BswModuleDescription			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
<b>Note</b>	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. <b>Tags:</b> atp.recommendedPackage=BswModuleDescriptions			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">AObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
bswModuleDependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=bswModuleDependency.shortName, bswModuleDependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
bswModuleDocumentation	<a href="#">SwComponentDocumentation</a>	0..1	aggr	This adds a documentation to the BSW module. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=bswModuleDocumentation, bswModuleDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6





Class	BswModuleDescription			
expectedEntry	<a href="#">BswModuleEntry</a>	*	ref	<p>Indicates an entry which is required by this module. Replacement of outgoingCallback / requiredEntry.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=expectedEntry.bswModuleEntry, expectedEntry.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime</p>
implementedEntry	<a href="#">BswModuleEntry</a>	*	ref	<p>Specifies an entry provided by this module which can be called by other modules. This includes "main" functions, interrupt routines, and callbacks. Replacement of providedEntry / expectedCallback.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=implementedEntry.bswModuleEntry, implementedEntry.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime</p>
internalBehavior	<a href="#">BswInternalBehavior</a>	*	aggr	<p>The various BswInternalBehaviors associated with a BswModuleDescription can be distributed over several physical files. Therefore the aggregation is &lt;&lt;atpSplitable&gt;&gt;.</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>  atp.Splitkey=internalBehavior.shortName  xml.sequenceOffset=65</p>
moduleId	PositiveInteger	0..1	attr	<p>Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen.</p> <p><b>Tags:</b>xml.sequenceOffset=5</p>
providedClientServerEntry	<a href="#">BswModuleClientServerEntry</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=providedClientServerEntry.shortName, providedClientServerEntry.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime  xml.sequenceOffset=45</p>
providedData	<a href="#">VariableDataPrototype</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>  atp.Splitkey=providedData.shortName, providedData.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime  xml.sequenceOffset=55</p>





Class	BswModuleDescription			
providedModeGroup	<a href="#">ModeDeclarationGroupPrototype</a>	*	aggr	<p>A set of modes which is owned and provided by this module or cluster. It can be connected to the required ModeGroups 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.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b>  atp.Splitkey=providedModeGroup.shortName, providedModeGroup.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime  xml.sequenceOffset=25</p>
releasedTrigger	<a href="#">Trigger</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b>  atp.Splitkey=releasedTrigger.shortName, releasedTrigger.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime  xml.sequenceOffset=35</p>
requiredClientServerEntry	BswModuleClientServerEntry	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b>  atp.Splitkey=requiredClientServerEntry.shortName, requiredClientServerEntry.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime  xml.sequenceOffset=50</p>
requiredData	<a href="#">VariableDataPrototype</a>	*	aggr	<p>Specifies a data prototype required by this module in order to be provided from another partition or core. The required Data 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.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b>  atp.Splitkey=requiredData.shortName, requiredData.variationPoint.shortLabel  vh.latestBindingTime=preCompileTime  xml.sequenceOffset=60</p>







Class	BswModuleDescription			
requiredModeGroup	<a href="#">ModeDeclarationGroup</a> <a href="#">Prototype</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=requiredModeGroup.shortName, requiredModeGroup.variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime                      xml.sequenceOffset=30</p>
requiredTrigger	<a href="#">Trigger</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=requiredTrigger.shortName, requiredTrigger.variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime                      xml.sequenceOffset=40</p>

**Table F.12: BswModuleDescription**

Class	BswModuleEntry			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
<b>Note</b>	<p>This class represents a single API entry (C-function prototype) into the BSW module or cluster.</p> <p>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.</p> <p><b>Tags:</b>atp.recommendedPackage=BswModuleEntries</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> <a href="#">Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
argument (ordered)	<a href="#">SwServiceArg</a>	*	aggr	<p>An argument belonging to this BswModuleEntry.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b>                      vh.latestBindingTime=blueprintDerivationTime                      xml.sequenceOffset=45</p>
bswEntryKind	BswEntryKindEnum	0..1	attr	<p>This describes whether the entry is concrete or abstract. If the attribute is missing the entry is considered as concrete.</p> <p><b>Tags:</b>xml.sequenceOffset=40</p>
callType	BswCallType	1	attr	<p>The type of call associated with this service.</p> <p><b>Tags:</b>xml.sequenceOffset=25</p>
executionContext	BswExecutionContext	1	attr	<p>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.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>
functionPrototypeEmitter	NameToken	0..1	attr	<p>This attribute is used to control the generation of function prototypes. If set to "RTE", the RTE generates the function prototypes in the Module Interlink Header File.</p>





Class	BswModuleEntry			
isReentrant	Boolean	1	attr	Reentrancy from the viewpoint of function callers: <ul style="list-style-type: none"> <li>• True: Enables the service to be invoked again, before the service has finished.</li> <li>• False: It is prohibited to invoke the service again before is has finished.</li> </ul> Tags:xml.sequenceOffset=15
isSynchronous	Boolean	1	attr	Synchronicity from the viewpoint of function callers: <ul style="list-style-type: none"> <li>• True: This calls a synchronous service, i.e. the service is completed when the call returns.</li> <li>• False: The service (on semantical level) may not be complete when the call returns.</li> </ul> Tags:xml.sequenceOffset=20
returnType	SwServiceArg	0..1	aggr	The return type belonging to this bswModuleEntry. Tags:xml.sequenceOffset=40
role	Identifier	0..1	attr	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 maybe vendor specific (e.g. for callbacks which can have more than one instance). Tags:xml.sequenceOffset=10
serviceld	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
swServiceImpl Policy	SwServiceImplPolicy Enum	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, MultilanguageReferrable, Referrable			
Attribute	Type	Mult.	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			





Class		BswServiceDependency		
Attribute	Type	Mult.	Kind	Note
assignedData	<a href="#">RoleBasedData Assignment</a>	*	aggr	Defines the role of an associated data object (owned by this module or cluster) in the context of the ServiceNeeds element.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
assignedEntry Role	RoleBasedBswModule EntryAssignment	*	aggr	Defines the role of an associated BswModuleEntry in the context of the ServiceNeeds element.  <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=assignedEntryRole, assignedEntry Role.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
ident	BswService DependencyIdent	0..1	aggr	This adds the ability to become referable to BswService Dependency.  <b>Stereotypes:</b> atpIdentityContributor <b>Tags:</b> xml.sequenceOffset=-100
serviceNeeds	<a href="#">ServiceNeeds</a>	1	aggr	The associated ServiceNeeds.

**Table F.15: BswServiceDependency**

Class		CompuConstFormulaContent		
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	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.			
<b>Base</b>	ARObject, CompuConstContent			
Attribute	Type	Mult.	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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=30

**Table F.16: CompuConstFormulaContent**

Class		DataMapping (abstract)		
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::DataMapping			
<b>Note</b>	Mapping of port elements (data elements and parameters) to frames and signals.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	ClientServerToSignalMapping, SenderReceiverCompositeElementToSignalMapping, SenderReceiverToSignalGroupMapping, <a href="#">SenderReceiverToSignalMapping</a> , TriggerToSignalMapping			
Attribute	Type	Mult.	Kind	Note
introduction	<a href="#">DocumentationBlock</a>	0..1	aggr	This represents introductory documentation about the data mapping.

**Table F.17: DataMapping**

<b>Class</b>	<b>Describable</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
<b>Note</b>	This meta-class represents the ability to add a descriptive documentation to non identifiable elements.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	CyclicTiming, EventControlledTiming, HwElementConnector, HwPinConnector, HwPinGroupConnector, I PduTiming, Ipv4DhcpServerConfiguration, Ipv6DhcpServerConfiguration, PncMapping, Socket Connection, <a href="#">TransformationComSpecProps</a> , <a href="#">TransformationDescription</a> , <a href="#">TransformationSignalProps</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
adminData	<a href="#">AdminData</a>	0..1	aggr	This represents the administrative data for the describable object.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=adminData xml.sequenceOffset=-20
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.  <b>Tags:</b> xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question.  More elaborate documentation, (in particular how the object is built or used) should go to "introduction".  <b>Tags:</b> xml.sequenceOffset=-60
introduction	<a href="#">DocumentationBlock</a>	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock.  <b>Tags:</b> xml.sequenceOffset=-30

**Table F.18: Describable**

<b>Class</b>	<<atpMixed>> <b>DocumentationBlock</b>			
<b>Package</b>	M2::MSR::Documentation::BlockElements			
<b>Note</b>	This class represents a documentation block. It is made of basic text structure elements which can be displayed in a table cell.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
defList	DefList	0..1	aggr	This represents a definition list in the documentation block.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=40
figure	MIFigure	0..1	aggr	This represents a figure in the documentation block.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=70





<b>Class</b>	<b>&lt;&lt;atpMixed&gt;&gt; DocumentationBlock</b>			
formula	MIFormula	0..1	aggr	This is a formula in the definition block. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=60
labeledList	LabeledList	0..1	aggr	This represents a labeled list. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=50
list	List	0..1	aggr	This represents numbered or unnumbered list. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=30
msrQueryP2	MsrQueryP2	0..1	aggr	This represents automatically contributed contents provided by an msrquery in the context of Documentation Block.
note	Note	0..1	aggr	This represents a note in the text flow. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=80
p	MultiLanguage Paragraph	0..1	aggr	This is one particular paragraph. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=10
structuredReq	StructuredReq	0..1	aggr	This aggregation supports structured requirements embedded in a documentation block. <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=90
verbatim	MultiLanguageVerbatim	0..1	aggr	This represents one particular verbatim text. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=20

**Table F.19: DocumentationBlock**

<b>Class</b>	<b>EcuInstance</b>
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology





<b>Class</b>		<b>EcuInstance</b>		
<b>Note</b>		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. <b>Tags:</b> atp.recommendedPackage=EcuInstances		
<b>Base</b>		<i>ARObject, CollectableElement, FibexElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>		
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
associatedComIPduGroup	ISignalIPduGroup	*	ref	With this reference it is possible to identify which ISignalIPduGroups are applicable for which Communication Connector/ ECU.  Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPduGroups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an Ecu Instance via the top level ISignalIPduGroup.
associatedConsumedProvidedServiceInstanceGroup	ConsumedProvidedServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
associatedPdurIPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PdurIPdu Groups are applicable for which Communication Connector/ 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_Main FunctionRouteSignals of the AUTOSAR COM module in seconds.
comConfigurationRxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionRx of the AUTOSAR COM module in seconds.
comConfigurationTxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionTx 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 cyclic Timing assigned or eventControlledTiming with numberOfRepetitions > 0).
commController	CommunicationController	1..*	aggr	CommunicationControllers of the ECU.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
connector	CommunicationConnector	*	aggr	All channels controlled by a single controller.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
dltConfig	DltConfig	0..1	aggr	Describes the Dlt configuration on this EcuInstance.
dolpConfig	DolpConfig	0..1	aggr	Dolp configuration on this EcuInstance.  <b>Tags:</b> atp.Status=draft
ecuTaskProxy	OsTaskProxy	*	ref	Reference to OsTaskProxies assigned to the Ecu Instance.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=ecuTaskProxy





Class	EcuInstance			
ethSwitchPortGroupDerivation	Boolean	0..1	attr	Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done.
partition	EcuPartition	*	aggr	Optional definition of Partitions within an Ecu.
pncPrepareSleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pncSynchronousWakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.
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.
sleepModeSupported	Boolean	1	attr	Specifies whether the ECU instance may be put to a "low power mode" <ul style="list-style-type: none"> <li>• true: sleep mode is supported</li> <li>• false: sleep mode is not supported</li> </ul> Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software.
tcplplcmpProps	EthTcplplcmpProps	0..1	ref	EcuInstance specific ICMP (Internet Control Message Protocol) attributes
tcplpProps	EthTcplpProps	0..1	ref	EcuInstance specific Tcplp Stack attributes.
v2xSupported	V2xSupportEnum	0..1	attr	This attribute is used to control the existence of the V2X stack on the given EcuInstance.
wakeUpOverBusSupported	Boolean	1	attr	Driver support for wakeup over Bus.

**Table F.20: EcuInstance**

Class	EcucValueCollection			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	This represents the anchor point of the ECU configuration description. Tags:atp.recommendedPackage=EcucValueCollections			
Base	AElement, AObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note
ecucValue	EcucModuleConfigurationValues	*	ref	References to the configuration of individual software modules that are present on this ECU. atpVariation: [RS_ECUC_00079] Stereotypes: atpVariation Tags:vh.latestBindingTime=preCompileTime
ecuExtract	System	0..1	ref	Represents the extract of the System Configuration that is relevant for the ECU configured with that ECU Configuration Description.

**Table F.21: EcucValueCollection**

<b>Class</b>	<b>EndToEndProtectionISignalIPdu</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::EndToEndProtection			
<b>Note</b>	<p>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 EndToEndProtectionISignalIPdu element shall be created within the EndToEndProtectionSet.</p> <p>The EndToEndProtectionISignalIPdu element refers to the ISignalGroup that is to be protected and to the ISignalIPdu 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.</p>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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	<a href="#">ISignalGroup</a>	1	ref	Reference to the ISignalGroup that is to be protected.
iSignalIPdu	<a href="#">ISignalIPdu</a>	1	ref	Reference to the ISignalIPdu that transmits the protected ISignalGroup.

**Table F.22: EndToEndProtectionISignalIPdu**

<b>Class</b>	<b>EndToEndTransformationDescription</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Transformer			
<b>Note</b>	EndToEndTransformationDescription holds these attributes which are profile specific and have the same value for all E2E transformers.			
<b>Base</b>	ARObject, <a href="#">Describable</a> , <a href="#">TransformationDescription</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
clearFromValidToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
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.
e2eProfileCompatibilityProps	<a href="#">E2EProfileCompatibilityProps</a>	0..1	ref	Reference to additional settings for the E2E state machine.
maxDeltaCounter	PositiveInteger	0..1	attr	Maximum allowed difference between two counter values of two consecutively received valid messages. For example, if the receiver gets data with counter 1 and MaxDeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
maxErrorStateInit	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size 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 Window Size checks, for the state E2E_SM_INVALID.
maxErrorStateValid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_VALID.
maxNoNewOrRepeatedData	PositiveInteger	0..1	attr	The maximum allowed amount of consecutive failed counter checks.







<b>Class</b>	<b>EndToEndTransformationDescription</b>			
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 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.
upperHeaderBitsToShift	PositiveInteger	0..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 shall be, due to compatibility between non-protected and E2E-protected communication, at the same position, which is before E2E header.
windowSizeInit	PositiveInteger	0..1	attr	Size of the monitoring window of state Init for the E2E state machine.
windowSizeInvalid	PositiveInteger	0..1	attr	Size of the monitoring window of state Invalid for the E2E state machine.
windowSizeValid	PositiveInteger	0..1	attr	Size of the monitoring window of state Valid for the E2E state machine.

**Table F.23: EndToEndTransformationDescription**

<b>Class</b>	<b>FlatInstanceDescriptor</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::FlatMap





<b>Class</b>		<b>FlatInstanceDescriptor</b>		
<b>Note</b>	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: <ul style="list-style-type: none"> <li>• Specify unique names of measurable data to be used by MCD tools</li> <li>• Specify unique names of calibration data to be used by MCD tool</li> <li>• Specify a unique name for an instance of a component prototype in the ECU extract of the system description</li> </ul> Note that in addition it is possible to assign alias names via AliasNameAssignment.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
ecuExtract Reference	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 Atomic SoftwareComponentType, which is referred by the particular SwcInternalBehavior.  <b>Tags:</b> xml.sequenceOffset=40 <b>InstanceRef implemented by:</b> <a href="#">AnyInstanceRef</a>
role	Identifier	0..1	attr	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. ModeDeclaration GroupPrototypes which are measurable. In this case the RTE will provide locations for current mode, previous mode and next mode.
rtePluginProps	RtePluginProps	0..1	aggr	The properties of a communication graph with respect to the utilization of RTE Implementation Plug-in.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=rtePluginProps
swDataDef Props	<a href="#">SwDataDefProps</a>	0..1	aggr	The properties of this FlatInstanceDescriptor.
upstream Reference	AtpFeature	0..1	iref	Refers to the instance in the context of an "upstream" descriptions, wich 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 Swc InternalBehavior.  <b>Tags:</b> xml.sequenceOffset=20 <b>InstanceRef implemented by:</b> <a href="#">AnyInstanceRef</a>

**Table F.24: FlatInstanceDescriptor**

<b>Class</b>	<b>HwDescriptionEntity</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::EcuResourceTemplate			
<b>Note</b>	This meta-class represents the ability to describe a hardware entity.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">HwElement</a> , <a href="#">HwPin</a> , <a href="#">HwPinGroup</a> , <a href="#">HwType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
hwAttribute Value	HwAttributeValue	*	aggr	This aggregation represents a particular hardware attribute value. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=systemDesignTime xml.sequenceOffset=50
hwCategory	HwCategory	*	ref	One of the associations representing one particular category of the hardware entity. <b>Tags:</b> xml.sequenceOffset=30
hwType	<a href="#">HwType</a>	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.25: HwDescriptionEntity**

<b>Class</b>	<b>HwElement</b>			
<b>Package</b>	M2::AUTOSARTemplates::EcuResourceTemplate			
<b>Note</b>	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. <b>Tags:</b> atp.recommendedPackage=HwElements			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">HwDescriptionEntity</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
hwElement Connection	HwElementConnector	*	aggr	This represents one particular connection between two hardware elements. <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=systemDesignTime xml.sequenceOffset=90





Class	HwElement			
nestedElement	<a href="#">HwElement</a>	*	ref	<p>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).</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=systemDesignTime xml.sequenceOffset=70</p>

**Table F.26: HwElement**

Class	HwType			
<b>Package</b>	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
<b>Note</b>	<p>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.</p> <p><b>Tags:</b>atp.recommendedPackage=HwTypes</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">HwDescriptionEntity</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table F.27: HwType**

Class	ISignal			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
<b>Note</b>	<p>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.</p> <p>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.</p> <p>ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping).</p> <p>In case of the SystemSignalGroup an ISignal shall be created for each SystemSignal contained in the SystemSignalGroup.</p> <p><b>Tags:</b>atp.recommendedPackage=ISignals</p>			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">FibexElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
data Transformation	<a href="#">DataTransformation</a>	0..1	ref	<p>Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=dataTransformation.dataTransformation, dataTransformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime</p>
dataTypePolicy	DataTypePolicyEnum	1	attr	<p>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</p>





Class	ISignal			
				<p>△</p> <p>used to support flexible development methodology as well as subsequent integrity checks.</p> <p>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.</p>
initValue	<a href="#">ValueSpecification</a>	0..1	aggr	<p>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.</p> <p>This value can be used to configure the Signal's "Init Value".</p> <p>If a full DataMapping exist for the SystemSignal this information may be available from a configured Sender ComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.</p>
iSignalProps	ISignalProps	0..1	aggr	<p>Additional optional ISignal properties that may be stored in different files.</p> <p><b>Stereotypes:</b> atpSplittable <b>Tags:</b>atp.Splitkey=iSignalProps</p>
iSignalType	ISignalTypeEnum	0..1	attr	<p>This attribute defines whether this iSignal is an array that results in a UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.</p>
length	UnlimitedInteger	1	attr	<p>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.</p> <p>The ISignal length of zero bits is allowed.</p>
network Representation Props	<a href="#">SwDataDefProps</a>	0..1	aggr	<p>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 "memAllignment" and "byteOrder" shall not be used.</p> <p>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.</p> <p>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.</p> <p>In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalid Value" and the Data Semantics.</p>
systemSignal	<a href="#">SystemSignal</a>	1	ref	<p>Reference to the System Signal that is supposed to be transmitted in the ISignal.</p>





Class	ISignal			
timeout Substitution Value	<a href="#">ValueSpecification</a>	0..1	aggr	Defines and enables the ComTimeoutSubstitution for this ISignal.
transformation ISignalProps	TransformationISignal Props	*	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.28: ISignal**

Class	ISignalGroup			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
<b>Note</b>	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)  <b>Tags:</b> atp.recommendedPackage=ISignalGroup			
<b>Base</b>	<i>ARObject, CollectableElement, FibexElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
comBased SignalGroup Transformation	<a href="#">DataTransformation</a>	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.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=comBasedSignalGroupTransformation.data Transformation, comBasedSignalGroup Transformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
iSignal	<a href="#">ISignal</a>	*	ref	Reference to a set of ISignals that shall always be kept together.
systemSignal Group	SystemSignalGroup	1	ref	Reference to the SystemSignalGroup that is defined on VFB level and that is supposed to be transmitted in the ISignalGroup.
transformation ISignalProps	TransformationISignal Props	*	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 ISignal Groups are described in the TransformationTechnology class.

**Table F.29: ISignalGroup**

Class	ISignalIPdu
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication





<b>Class</b>	<b>ISignalIPdu</b>			
<b>Note</b>	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.  <b>Tags:</b> atp.recommendedPackage=Pdus			
<b>Base</b>	ARObject, CollectableElement, FibexElement, IPdu, Identifiable, MultilanguageReferrable, PackageableElement, Pdu, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
iSignalToPdu Mapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the Signal IPdu.  atpVariation: The content of a PDU can be variable.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
unusedBit Pattern	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.30: ISignalIPdu**

<b>Class</b>	<b>Identifiable</b> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
<b>Note</b>	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.
<b>Base</b>	ARObject, MultilanguageReferrable, Referrable
<b>Subclasses</b>	ARPackage, AbstractDolpLogicAddressProps, AbstractEvent, AbstractImplementationDataTypeElement, AbstractSecurityEventFilter, AbstractSecurityIdsmInstanceFilter, AbstractServiceInstance, AppOsTaskProxyToEcuTaskProxyMapping, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartitionMapping, AsynchronousServerCallResultPoint, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpFeature, AutosarOperationArgumentInstance, AutosarVariableInstance, BinaryManifestAddressableObject, BinaryManifestItemDefinition, BinaryManifestResource, BinaryManifestResourceDefinition, BlockState, BswInternalTriggeringPoint, BswModuleDependency, BuildActionEntity, BuildActionEnvironment, CanTpAddress, CanTpChannel, CanTpNode, Chapter, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, CollectableElement, ComManagementMapping, CommConnectorPort, CommunicationConnector, CommunicationController, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, CouplingPortStructuralElement, CpSoftwareClusterResource, CpSoftwareClusterResourceToApplicationPartitionMapping, CpSoftwareClusterToEcuInstanceMapping, CpSoftwareClusterToResourceMapping, CryptoServiceMapping, DataPrototypeGroup, DataTransformation, DependencyOnArtifact, DiagEventDebounceAlgorithm, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, DiagnosticFunctionInhibitSource, DiagnosticRoutineSubfunction, DitApplication, DitArgument, DitLogChannel, DitMessage, DolpInterface, DolpLogicAddress, DolpRoutingActivation, ECUMapping, EOExecutableEntityRefAbstract, EcuPartition, EcuContainerValue, EcuDefinitionElement, EcuDestinationUriDef, EcuEnumerationLiteralDef, EcuQuery, EcuValidationCondition, EndToEndProtection, EthernetWakeupSleepOnDatalineConfig, EventHandler, ExclusiveArea, ExecutableEntity, ExecutionTime, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FlatInstance





Class	Identifiable (abstract)			
	<p style="text-align: center;">△</p> <p>Descriptor, FlexrayArTpNode, FlexrayTpConnectionControl, FlexrayTpNode, FlexrayTpPduPool, Frame Triggering, GeneralParameter, GlobalTimeGateway, GlobalTimeMaster, GlobalTimeSlave, HeapUsage, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, IdentCaption, InternalTriggeringPoint, J1939SharedAddressCluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, MacMulticastGroup, McDataInstance, MemorySection, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, Network Endpoint, NmCluster, NmEcu, NmNode, NvBlockDescriptor, PackageableElement, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, PhysicalChannel, PortElementToCommunicationResourceMapping, PortGroup, PortInterfaceMapping, PossibleErrorReaction, ResourceConsumption, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RteEventInCompositionSeparation, RteEventInCompositionToOsTaskProxyMapping, RteEventInSystemSeparation, RteEventInSystemToOsTaskProxyMapping, RunnableEntityGroup, SdgAttribute, SdgClass, SecureCommunicationAuthenticationProps, SecureCommunicationFreshnessProps, SecurityEventContextProps, ServerCallPoint, ServiceNeeds, SignalServiceTranslationElementProps, SignalServiceTranslationEventProps, SignalServiceTranslationProps, SocketAddress, SomeipTpChannel, SpecElementReference, StackUsage, StaticSocketConnection, StructuredReq, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImpMapping, SystemMapping, TDCpSoftwareClusterMapping, TDCpSoftwareClusterResourceMapping, TcpOptionFilterList, TimingCondition, TimingConstraint, TimingDescription, TimingExtensionResource, TimingModelInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, Topic1, TpAddress, TraceableTable, TraceableText, TracedFailure, TransformationProps, TransformationTechnology, Trigger, VariableAccess, VariationPointProxy, ViewMap, VlanConfig, WaitPoint</p>			
Attribute	Type	Mult.	Kind	Note
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=adminData xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes.  <b>Tags:</b> xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints.  <b>Tags:</b> xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question.  More elaborate documentation, (in particular how the object is built or used) should go to "introduction".  <b>Tags:</b> xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock.  <b>Tags:</b> xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID







<b>Class</b>	<b>Identifiable</b> (abstract)			
				<p>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.</p> <p><b>Tags:</b>xml.attribute=true</p>

**Table F.31: Identifiable**

<b>Class</b>	<b>InstantiationTimingEventProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	This meta-class represents the ability to refine a timing event for particular instances of a software component. This approach supports an instance specific timing.			
<b>Base</b>	ARObject, <a href="#">InstantiationRTEEventProps</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
period	<a href="#">TimeValue</a>	0..1	attr	This attribute represents the value of the refined activation period.

**Table F.32: InstantiationTimingEventProps**

<b>Class</b>	<b>McDataInstance</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport			
<b>Note</b>	<p>Describes the specific properties of one data instance in order to support measurement and/or calibration of this data instance.</p> <p>The most important attributes are:</p> <ul style="list-style-type: none"> <li>• Its shortName is copied from the ECU Flat map (if applicable) and will be used as identifier and for display by the MC system.</li> <li>• The category is copied from the corresponding data type (ApplicationDataType if defined, otherwise ImplementationDataType) as far as applicable.</li> <li>• 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.</li> </ul> <p>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.</p>			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.





Class	McDataInstance			
displayIdentifier	McIdIdentifier	0..1	attr	An optional attribute to be used to set the ASAM ASAP2 DISPLAY_IDENTIFIER attribute.
flatMapEntry	<a href="#">FlatInstanceDescriptor</a>	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 <ul style="list-style-type: none"> <li>• The McDataInstance may represent an array or struct in which only the subElements correspond to FlatMap entries.</li> <li>• The McDataInstance may represent a task local buffer for rapid prototyping access which is different from the "main instance" used for measurement access.</li> </ul>
instanceInMemory	ImplementationElement InParameterInstance Ref	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	RoleBasedMcData Assignment	*	aggr	An assignment between McDataInstances. This supports the indication of related McDataElement implementing the of "RP global buffer", "RP global measurement buffer", "RP enabler flag".
resultingProperties	<a href="#">SwDataDefProps</a>	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.
resultingRptSwPrototypingAccess	<a href="#">RptSwPrototyping Access</a>	0..1	aggr	Describes the implemented accessibility of data and modes by the rapid prototyping tooling.
role	Identifier	0..1	attr	An optional attribute to be used for additional information on the role of this data instance, for example in the context of rapid prototyping.
rptImplPolicy	<a href="#">RptImplPolicy</a>	0..1	aggr	Describes the implemented code preparation for rapid prototyping at data accesses for a hook based bypassing.
subElement (ordered)	<a href="#">McDataInstance</a>	*	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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
symbol	SymbolString	0..1	attr	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.





<b>Class</b>	<b>McDataInstance</b>			
				<p style="text-align: center;">△</p> <p>The symbol can differ from the shortName in case of generated C data declarations.</p> <p>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.</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>atp.Splitkey=symbol</p>

**Table F.33: McDataInstance**

<b>Class</b>	<b>McSupportData</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
emulation Support	McSwEmulationMethod Support	*	aggr	<p>Describes the calibration method used by the RTE. This information is not needed for A2L generation, but to setup software emulation in the ECU.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b>vh.latestBindingTime=preCompileTime</p>
mcParameter Instance	<a href="#">McDataInstance</a>	*	aggr	<p>A data instance to be used for calibration.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=mcParameterInstance.shortName, mcParameterInstance.variationPoint.shortLabel                      vh.latestBindingTime=postBuild</p>
mcVariable Instance	<a href="#">McDataInstance</a>	*	aggr	<p>A data instance to be used for measurement.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=mcVariableInstance.shortName, mcVariableInstance.variationPoint.shortLabel                      vh.latestBindingTime=postBuild</p>
measurable System ConstantValues	<a href="#">SwSystemconstant ValueSet</a>	*	ref	<p>Sets of system constant values to be transferred to the MCD system, because the system constants have been specified with "swCalibrationAccess" = readonly.</p>
rptSupportData	RptSupportData	0..1	aggr	<p>The rapid prototyping support data belonging to this implementation. The aggregation is &lt;&lt;atpSplitable&gt;&gt; because in case of an already existing BSW Implementation model, this description will be added later in the process, namely at code generation time.</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>atp.Splitkey=rptSupportData</p>

**Table F.34: McSupportData**

<b>Class</b>	<b>MultilanguageReferrable</b> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable





<b>Class</b>	<b>MultilanguageReferrable</b> (abstract)			
<b>Note</b>	Instances of this class can be referred to by their identifier (while adhering to namespace borders). They also may have a longName. But they are not considered to contribute substantially to the overall structure of an AUTOSAR description. In particular it does not contain other Referrables.			
<b>Base</b>	ARObject, Referrable			
<b>Subclasses</b>	Caption, DefItem, DocumentationContext, Identifiable, SdgCaption, TraceReferrable, Traceable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
longName	MultilanguageLong Name	0..1	aggr	This specifies the long name of the object. Long name is targeted to human readers and acts like a headline.

**Table F.35: MultilanguageReferrable**

<b>Class</b>	<b>ParameterInAtomicSWCTypeInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements::InstanceRefs Usage			
<b>Note</b>	This class implements an instance reference which can be applied for variables as well as for parameters.			
<b>Base</b>	ARObject, AtpInstanceRef			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	AtomicSwComponent Type	0..1	ref	<b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
contextData Prototype (ordered)	ApplicationComposite ElementDataPrototype	*	ref	This ist the context in a compositeDataType. <b>Tags:</b> xml.sequenceOffset=40
portPrototype	PortPrototype	0..1	ref	This is the port providing the variable or the entry point to the variable structure. <b>Tags:</b> xml.sequenceOffset=20
rootParameter DataPrototype	DataPrototype	0..1	ref	This represents the entry point for references into a CompositeDataType. <b>Tags:</b> xml.sequenceOffset=30
targetData Prototype	DataPrototype	0..1	ref	This is the target parameter element. Note that this must be nested in ParameterDataPrototype. The target must be one of ParameterDataPrototype, Application CompositeElementDataPrototype. <b>Tags:</b> xml.sequenceOffset=50

**Table F.36: ParameterInAtomicSWCTypeInstanceRef**

<b>Class</b>	<b>PostBuildVariantCriterionValueSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	This meta-class represents the ability to denote one set of postBuildVariantCriterionValues. <b>Tags:</b> atp.recommendedPackage=PostBuildVariantCriterionValueSets			
<b>Base</b>	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
postBuildVariant CriterionValue	PostBuildVariant CriterionValue	*	aggr	This is is one particular postbuild variant criterion/value pair being part of the PostBuildVariantSet.

**Table F.37: PostBuildVariantCriterionValueSet**

<b>Class</b>	<b>Referrable</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
<b>Note</b>	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
<b>Base</b>	ARObject			
<b>Subclasses</b>	AtpDefinition, BswDistinguishedPartition, BswModuleCallPoint, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, DiagnosticEnvModeElement, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, HwDescriptionEntity, ImplementationProps, LinSlaveConfigIdent, ModeTransition, MultilanguageReferrable, PncMappingIdent, SingleLanguageReferrable, SoConlPdulIdentifier, SocketConnectionBundle, TimeSyncServerConfiguration, TpConnectionIdent			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference.  <b>Stereotypes:</b> atpIdentityContributor <b>Tags:</b> xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments.  <b>Tags:</b> xml.sequenceOffset=-90

**Table F.38: Referrable**

<b>Class</b>	<b>RootSwCompositionPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate			
<b>Note</b>	<p>The RootSwCompositionPrototype represents the top-level-composition of software components within a given System.</p> <p>According to the use case of the System, this may for example be 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.</p> <p>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.</p> <p>The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.</p>			
<b>Base</b>	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
calibration ParameterValue Set	CalibrationParameter ValueSet	*	ref	Used CalibrationParameterValueSet for instance specific initialization of calibration parameters.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=calibrationParameterValueSet
flatMap	FlatMap	0..1	ref	The FlatMap used in the scope of this RootSw CompositionPrototype.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=flatMap
software Composition	CompositionSw ComponentType	1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system.  <b>Stereotypes:</b> isOfType

**Table F.39: RootSwCompositionPrototype**

<b>Class</b>	<b>Sdg</b>			
<b>Package</b>	M2::MSR::AsamHdo::SpecialData			
<b>Note</b>	<p>Sdg (SpecialDataGroup) is a generic model which can be used to keep arbitrary information which is not explicitly modeled in the meta-model.</p> <p>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.</p> <p>Thereby SDG should be considered as a temporary solution when no explicit model is available. If an sdg Caption is available, it is possible to establish a reference to the sdg structure.</p>			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
gid	NameToken	1	attr	<p>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.</p> <p><b>Tags:</b>xml.attribute=true</p>
sdgCaption	SdgCaption	0..1	aggr	<p>This aggregation allows to assign the properties of Identifiable to the sdg. By this, a shortName etc. can be assigned to the Sdg.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>
sdgContentsType	SdgContents	0..1	aggr	<p>This is the content of the Sdg.</p> <p><b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false</p>

**Table F.40: Sdg**

<b>Class</b>	<b>SenderReceiverToSignalMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::DataMapping			
<b>Note</b>	Mapping of a sender receiver communication data element to a signal.			
<b>Base</b>	ARObject, <a href="#">DataMapping</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataElement	<a href="#">VariableDataPrototype</a>	1	iref	<p>Reference to the data element.</p> <p><b>InstanceRef implemented by:</b>VariableDataPrototypeIn SystemInstanceRef</p>
senderToSignal TextTable Mapping	<a href="#">TextTableMapping</a>	0..1	aggr	<p>This mapping allows for the text-table translation between the sending DataPrototype that is defined in the Port Prototype and the physicalProps defined for the System Signal.</p>
signalTo ReceiverText TableMapping	<a href="#">TextTableMapping</a>	0..1	aggr	<p>This mapping allows for the text-table translation between the physicalProps defined for the SystemSignal and a receiving DataPrototype that is defined in the Port Prototype.</p>
systemSignal	<a href="#">SystemSignal</a>	1	ref	<p>Reference to the system signal used to carry the data element.</p>

**Table F.41: SenderReceiverToSignalMapping**

<b>Primitive</b>	<b>String</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
<b>Note</b>	<p>This represents a String in which white-space shall be normalized before processing. For example: in order to compare two Strings:</p> <ul style="list-style-type: none"> <li>• leading and trailing white-space needs to be removed</li> <li>• consecutive white-space (blank, cr, lf, tab) needs to be replaced by one blank.</li> </ul> <p><b>Tags:</b> xml.xsd.customType=STRING xml.xsd.type=string</p>

**Table F.42: String**

<b>Class</b>	<b>SwServiceArg</b>			
<b>Package</b>	M2::MSR::DataDictionary::ServiceProcessTask			
<b>Note</b>	<p>Specifies the properties of a data object exchanged during the call of an SwService, e.g. an argument or a return value.</p> <p>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.</p>			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
direction	<a href="#">ArgumentDirection Enum</a>	0..1	attr	<p>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.</p> <p>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".</p> <p><b>Tags:</b>xml.sequenceOffset=10</p>
swArraysSize	<a href="#">ValueList</a>	0..1	aggr	<p>This turns the argument of the service to an array.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>
swDataDef Props	<a href="#">SwDataDefProps</a>	0..1	aggr	<p>Data properties of this SwServiceArg.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>

**Table F.43: SwServiceArg**

<b>Class</b>	<<atpMixedString>> <b>SwSystemconstDependentFormula</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	This class represents an expression depending on system constants.			
<b>Base</b>	ARObject, <a href="#">FormulaExpression</a>			
<b>Subclasses</b>	<a href="#">AttributeValueVariationPoint</a> , <a href="#">BlueprintFormula</a> , <a href="#">ConditionByFormula</a> , <a href="#">FMFormulaByFeaturesAndSwSystemconsts</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
sysc	<a href="#">SwSystemconst</a>	0..1	ref	<p>This refers to a system constant. The internal (coded) value of the system constant shall be used.</p> <p><b>Tags:</b>xml.sequenceOffset=50</p>





<b>Class</b>	<<atpMixedString>> <b>SwSystemconstDependentFormula</b> (abstract)			
syscString	<a href="#">SwSystemconst</a>	0..1	ref	syscString indicates that the referenced system constant shall be evaluated as a string according to [TPS_SWCT_01431].

**Table F.44: SwSystemconstDependentFormula**

<b>Class</b>	<b>SwSystemconstantValueSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	This meta-class represents the ability to specify a set of system constant values. <b>Tags:</b> atp.recommendedPackage=SwSystemconstantValueSets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
sw Systemconstant Value	SwSystemconstValue	*	aggr	This is one particular value of a system constant.

**Table F.45: SwSystemconstantValueSet**

<b>Class</b>	<b>System</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate			
<b>Note</b>	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. <b>Tags:</b> atp.recommendedPackage=Systems			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
clientId DefinitionSet	ClientIdDefinitionSet	*	ref	Set of Client Identifiers that are used for inter-ECU client-server communication in the System.
containerIPdu HeaderByte Order	<a href="#">ByteOrderEnum</a>	0..1	attr	Defines the byteOrder of the header in ContainerIPdus.
ecuExtract Version	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 Fibex Elements can be optional. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
interpolation Routine MappingSet	<a href="#">InterpolationRoutine MappingSet</a>	*	ref	This reference identifies the InterpolationRoutineMapping Sets that are relevant in the context of the enclosing System.







Class	System			
j1939SharedAddressCluster	J1939SharedAddressCluster	*	aggr	Collection of J1939Clusters that share a common address space for the routing of messages. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=j1939SharedAddressCluster.shortName, j1939SharedAddressCluster.variationPoint.shortLabel 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 atpSplitable and atpVariation. The content of System Mapping 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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=mapping.shortName, mapping.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	<a href="#">RootSwComposition Prototype</a>	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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=rootSoftwareComposition.shortName, rootSoftwareComposition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
swCluster	CpSoftwareCluster	*	ref	CP Software Clusters of this System <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=swCluster.cpSoftwareCluster, swCluster.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=systemDesignTime
systemDocumentation	<a href="#">Chapter</a>	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=systemDocumentation.shortName, systemDocumentation.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10
systemVersion	RevisionLabelString	1	attr	Version number of the System Description.

**Table F.46: System**

<b>Class</b>	<b>SystemSignal</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
<b>Note</b>	<p>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.</p> <p><b>Tags:</b>atp.recommendedPackage=SystemSignals</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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	<a href="#">SwDataDefProps</a>	0..1	aggr	Specification of the physical representation.

**Table F.47: SystemSignal**

<b>Class</b>	<b>TransientFault</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	The reported failure is classified as runtime error.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">TracedFailure</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
possibleError Reaction	PossibleErrorReaction	*	aggr	Describes a possible error reactions for the transient fault handler.

**Table F.48: TransientFault**

<b>Class</b>	<b>VariableInAtomicSWCTypeInstanceRef</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements::InstanceRefs Usage			
<b>Note</b>				
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpInstanceRef</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	<a href="#">AtomicSwComponentType</a>	0..1	ref	<b>Stereotypes:</b> atpDerived <b>Tags:</b> xml.sequenceOffset=10
contextData Prototype (ordered)	<a href="#">ApplicationCompositeElementDataPrototype</a>	*	ref	This is the context in a compositeDataType. <b>Tags:</b> xml.sequenceOffset=40
portPrototype	<a href="#">PortPrototype</a>	0..1	ref	This is the port providing the parameter or the entry point to the parameter structure. <b>Tags:</b> xml.sequenceOffset=20
rootVariable DataPrototype	<a href="#">VariableDataPrototype</a>	0..1	ref	<b>Tags:</b> xml.sequenceOffset=30
targetData Prototype	<a href="#">DataPrototype</a>	0..1	ref	This is the target of the instance ref. Note that it shall be one of ApplicationCompositeElementDataPrototype of VariableDataPrototype. <b>Tags:</b> xml.sequenceOffset=50

**Table F.49: VariableInAtomicSWCTypeInstanceRef**

<b>Class</b>	<b>VariationPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	This meta-class represents the ability to express a "structural variation point". The container of the variation point is part of the selected variant if swSyscond evaluates to true and each postBuildVariant Criterion is fulfilled.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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. <b>Tags:</b> xml.sequenceOffset=28
desc	MultiLanguageOverviewParagraph	0..1	aggr	This allows to describe shortly the purpose of the variation point. <b>Tags:</b> xml.sequenceOffset=20
formalBlueprintGenerator	BlueprintGenerator	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint by using ARML.  Note that variationPoints are not allowed within a formalBlueprintGenerator. <b>Tags:</b> atp.Status=draft xml.sequenceOffset=30
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. <b>Tags:</b> xml.sequenceOffset=40
sdg	Sdg	0..1	aggr	An optional special data group is attached to every variation point. These data can be used by external software systems to attach application specific data. For example, a variant management system might add an identifier, an URL or a specific classifier. <b>Tags:</b> xml.sequenceOffset=50
shortLabel	Identifier	0..1	attr	This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting 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. <b>Stereotypes:</b> atpIdentityContributor <b>Tags:</b> xml.sequenceOffset=10
swSyscond	ConditionByFormula	0..1	aggr	This condition acts as Binding Function for the Variation Point. Note that the multiplicity is 0..1 in order to support pure postBuild variants. <b>Tags:</b> xml.sequenceOffset=30

**Table F.50: VariationPoint**

## 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 (in this case: Software Component 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 upstream mapping tables contain the following columns:

Column Name	Column Meaning
BSW Module	Name of BSW module
BSW Context	Reference to parameter container
BSW Parameter	Name of the BSW parameter
BSW Type	Type of parameter
BSW Description	Description from the configuration document
Template Description	Class note or attribute note of the M2 model element
M2 Parameter	Name of the upstream template model element
Mapping Rule	Textual description on how to transform between M2 and BSW domains
Mapping Type	One of: <b>local</b> no mapping needed since parameter local to BSW <b>partial</b> some data can be automatically mapped but not all <b>full</b> all data can be automatically mapped
Mapping Status	Indication of life-cycle status of the mapping
ECUC Parameter ID	ID of the parameter in the respective SWS document (may be empty if the mapping is owned by an enumeration literal)

**Table G.1: Upstream mapping table columns**

### G.2 NvM

BSW Module	BSW Context
NvM	NvM/NvMBlockDescriptor
BSW Parameter	BSW Type
NvMBlockJobPriority	ECUC-INTEGER-PARAM-DEF





<b>BSW Description</b>	
Defines the job priority for a NVRAM block (0 = Immediate priority).	
<b>Template Description</b>	
<p><b>NvBlockNeeds.writingPriority:</b> Requires the priority of writing this block in case of concurrent requests to write other blocks.</p> <p><b>NvBlockNeeds.storeEmergency:</b> 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.</p>	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::NvBlockNeeds.writingPriority, CommonStructure::ServiceNeeds::NvBlockNeeds.storeEmergency	
<b>Mapping Rule</b>	<b>Mapping Type</b>
It is the integrators job to secure the value-monotonic assignment of writingPriority to NvMBlockJobPriority. This means that the lowest assigned value of writingPriority=MEDIUM shall be greater than highest assigned value of writingPriority=HIGH etc.If NvBlockNeeds.storeEmergency is set to True then NvMBlockJobPriority shall be 0 (Immediate priority). If NvBlockNeeds.storeEmergency is set to False then the value of NvMBlockJobPriority depends on the value of NvBlockNeeds.writingPriority.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_NvM_00477]

<b>BSW Module</b>	<b>BSW Context</b>	
NvM	NvM/NvMBlockDescriptor	
<b>BSW Parameter</b>		<b>BSW Type</b>
NvMBlockManagementType		ECUC-ENUMERATION-PARAM-DEF
<b>BSW Description</b>		
Defines the block management type for the NVRAM block.[SWS_NvM_00137]		
<b>Template Description</b>		
Reliability against data loss on the non-volatile medium.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::NvBlockNeeds.reliability		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
if (reliability == errorDetection   noProtection) && nDataSets==0 then NvmBlockManagementType = NVM_BLOCK_NATIVE. if reliability == errorCorrection then NvmBlockManagementType = NVM_BLOCK_REDUNDANT. [constr_1095] applies.	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_NvM_00062]	

<b>BSW Module</b>	<b>BSW Context</b>	
NvM	NvM/NvMBlockDescriptor	
<b>BSW Parameter</b>		<b>BSW Type</b>
NvMBlockUseAutoValidation		ECUC-BOOLEAN-PARAM-DEF
<b>BSW Description</b>		
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	ECUC Parameter ID
valid	[ECUC_NvM_00557]

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockUseCRCCompMechanism		ECUC-BOOLEAN-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_NvM_00556]

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockUseCrc		ECUC-BOOLEAN-PARAM-DEF
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 bet set to true, else NvmBlockUseCrc = false		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_NvM_00036]

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockUseSetRamBlockStatus		ECUC-BOOLEAN-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_NvM_00552]

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockWriteProt		ECUC-BOOLEAN-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_NvM_00033]

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMCalcRamBlockCrc		ECUC-BOOLEAN-PARAM-DEF
BSW Description		





<p>Defines CRC (re)calculation for the permanent RAM block or NVRAM blocks which are configured to use explicit synchronization mechanism.</p> <p>true: CRC will be (re)calculated for this permanent RAM block. false: CRC will not be (re)calculated for this permanent RAM block.</p>	
<b>Template Description</b>	
Defines if CRC (re)calculation for the permanent RAM Block is required.	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::NvBlockNeeds.calcRamBlockCrc	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_NvM_00119]

<b>BSW Module</b>	<b>BSW Context</b>	
NvM	NvM/NvMBlockDescriptor	
<b>BSW Parameter</b>		<b>BSW Type</b>
NvMNvBlockNum		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
<p>Defines the number of multiple NV blocks in a contiguous area according to the given block management type.</p> <p>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.</p> <p>1 For NVRAM blocks to be configured of block management type NVM_BLOCK_NATIVE</p> <p>2 For NVRAM blocks to be configured of block management type NVM_BLOCK_REDUNDANT</p>		
<b>Template Description</b>		
<p><b>NvBlockNeeds.nDataSets:</b> 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.</p> <p><b>NvBlockNeeds.reliability:</b> Reliability against data loss on the non-volatile medium.</p>		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::NvBlockNeeds.nDataSets, CommonStructure::ServiceNeeds::NvBlockNeeds.reliability		
<b>Mapping Rule</b>		<b>Mapping Type</b>
if (nDataSets == 0 && reliability ==noProtection   errorDetection) then NvMNvBlockNum = 1. if (nDataSets >0 && reliability ==noProtection   errorDetection) then NvMNvBlockNum = nDataSets.		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_NvM_00480]

<b>BSW Module</b>	<b>BSW Context</b>	
NvM	NvM/NvMBlockDescriptor	
<b>BSW Parameter</b>		<b>BSW Type</b>
NvMResistantToChangedSw		ECUC-BOOLEAN-PARAM-DEF
<b>BSW Description</b>		







<p>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 <code>NvM_GetErrorStatus()</code> to be able to distinguish between first initialization and corrupted data.</p> <p>true: NVRAM block is resistant to changed software. false: NVRAM block is not resistant to changed software.</p>	
<b>Template Description</b>	
<p>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.</p>	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::NvBlockNeeds.resistantToChangedSw	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 Mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_NvM_00483]

<b>BSW Module</b>	<b>BSW Context</b>	
NvM	NvM/NvMBlockDescriptor	
<b>BSW Parameter</b>		<b>BSW Type</b>
NvMRomBlockNum		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
<p>Defines the number of multiple ROM blocks in a contiguous area according to the given block management type.</p> <p>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.</p> <p>0-1 For NVRAM blocks to be configured of block management type NVM_BLOCK_NATIVE</p> <p>0-1 For NVRAM blocks to be configured of block management type NVM_BLOCK_REDUNDANT</p>		
<b>Template Description</b>		
<p>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.</p>		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::NvBlockNeeds.nRomBlocks		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_NvM_00485]	

<b>BSW Module</b>	<b>BSW Context</b>	
NvM	NvM/NvMBlockDescriptor	
<b>BSW Parameter</b>		<b>BSW Type</b>
NvMSelectBlockForFirstInitAll		ECUC-BOOLEAN-PARAM-DEF
<b>BSW Description</b>		
<p>Defines whether a block will be processed or not by <code>NvM_FirstInitAll</code>. A block can be configured to be processed even if it doesn't have permanent RAM and/or explicit synchronization.</p> <p>TRUE: block will be processed by <code>NvM_FirstInitAll</code></p> <p>FALSE: block will not be processed by <code>NvM_FirstInitAll</code></p>		
<b>Template Description</b>		





If this attribute is set to true the NvM shall process this block in the NvM_FirstInitAll() function.	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::NvBlockNeeds. <a href="#">selectBlockForFirstInitAll</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_NvM_00558]

<b>BSW Module</b>	<b>BSW Context</b>	
NvM	NvM/NvMBlockDescriptor	
<b>BSW Parameter</b>		<b>BSW Type</b>
NvMSelectBlockForReadAll		ECUC-BOOLEAN-PARAM-DEF
<b>BSW Description</b>		
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		
<b>Template Description</b>		
Defines whether the associated RAM Block shall be implicitly restored during startup by the basic software.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::NvBlockNeeds. <a href="#">restoreAtStart</a>		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 Mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_NvM_00117]

<b>BSW Module</b>	<b>BSW Context</b>	
NvM	NvM/NvMBlockDescriptor	
<b>BSW Parameter</b>		<b>BSW Type</b>
NvMSelectBlockForWriteAll		ECUC-BOOLEAN-PARAM-DEF
<b>BSW Description</b>		
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		
<b>Template Description</b>		
Defines whether or not the associated RAM Block shall be implicitly stored during shutdown by the basic software.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::NvBlockNeeds. <a href="#">storeAtShutdown</a>		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 Mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_NvM_00549]

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter	BSW Type	
NvMStaticBlockIDCheck	ECUC-BOOLEAN-PARAM-DEF	
BSW Description		
<p>Defines if the Static Block ID check is enabled. false: Static Block ID check is disabled. true: Static Block ID check is enabled.</p>		
Template Description		
<p>Defines if the Static Block Id check shall be enabled.</p>		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds. <a href="#">checkStaticBlockId</a>		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_NvM_00532]

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter	BSW Type	
NvMWriteBlockOnce	ECUC-BOOLEAN-PARAM-DEF	
BSW Description		
<p>Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. true: Defines write protection after first write is enabled. false: Defines write protection after first write is disabled.</p>		
Template Description		
<p>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.</p>		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds. <a href="#">writeOnlyOnce</a>		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_NvM_00072]

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter	BSW Type	
NvMWriteVerification	ECUC-BOOLEAN-PARAM-DEF	
BSW Description		





Defines if Write Verification is enabled. false: Write verification is disabled. true: Write Verification is enabled.	
<b>Template Description</b>	
Defines if Write Verification shall be enabled for this NVRAM Block.	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::NvBlockNeeds.writeVerification	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_NvM_00534]

### G.3 Com

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilter		ECUC-PARAM-CONF-CONTAINER-DEF
<b>BSW Description</b>		
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.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter		
<b>Mapping Rule</b>		<b>Mapping Type</b>
Create container on the receiver side if the NonqueuedReceiverComSpec contains a DataFilter. Create Container on the sender side if the TransmissionModeCondition element contains a reference to this signal.		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Com_00339]

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilterAlgorithm		ECUC-ENUMERATION-PARAM-DEF
<b>BSW Description</b>		
The range of values is specified in the [17] specification, chapter 2.2.2, Reception Filtering.		
<b>Template Description</b>		
This attribute specifies the type of the filter.		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.dataFilterType		
<b>Mapping Rule</b>		<b>Mapping Type</b>





Mapping between DataFilterTypeEnum and ComFilterAlgorithm Enum is necessary.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Com_00146]

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilterMask		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
<b>Template Description</b>		
Mask for old and new value.		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.mask		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Com_00235]

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilterMax		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
<b>Template Description</b>		
Value to specify the upper boundary		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.max		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Com_00317]

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilterMin		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
<b>Template Description</b>		





Value to specify the lower boundary	
<b>M2 Parameter</b>	
CommonStructure::Filter::DataFilter.min	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Com_00318]

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilterOffset		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering. Range = 0..(ComFilterPeriod-1)		
<b>Template Description</b>		
Specifies the initial number of messages to occur before the first message is passed		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.offset		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Com_00313]	

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilterPeriod		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
This parameter defines the period of the ComFilterAlgorithm ONE_EVERY_N.		
<b>Template Description</b>		
Specifies number of messages to occur before the message is passed again		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.period		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Com_00312]	

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	





BSW Parameter		BSW Type
ComFilterX		ECUC-INTEGER-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_Com_00147]

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription	
BSW Parameter		BSW Type
ComSignalInitValue		ECUC-STRING-PARAM-DEF
BSW Description		
<p>Initial value for this signal. In case of UIN8_N the default value is a string of length ComSignalLength with all bytes set to 0x00. In case of UIN8_DYN the initial size shall be 0.</p> <p>In case the ComSignalType is UIN8, UIN16, UIN32, UIN64, SIN8, SIN16, SIN32, SIN64 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 UIN8_N, UIN8_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 UIN8_DYN the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal.</p>		
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, SWComponent Template::Communication::NonqueuedSenderComSpec.initValue		
Mapping Rule		Mapping Type
It is possible to aggregate an initValue at the level of a ComSpec in the SWC Template. in case the System Description doesn't use a complete Software Component Description (VFB View) the init Value is defined in the System Template.		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Com_00170]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal	
BSW Parameter		BSW Type
ComDataInvalidAction		ECUC-ENUMERATION-PARAM-DEF
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.	
<b>Template Description</b>	
<p><b>InvalidationPolicy:</b> Specifies whether the component can actively invalidate a particular dataElement.</p> <p>If no invalidationPolicy points to a dataElement this is considered to yield the identical result as if the handleInvalid attribute was set to dontInvalidate.</p> <p><b>ISignalPort.handleInvalid:</b> This attribute defines how invalidation is applied to the ISignals received in the context of this ISignalPort.</p>	
<b>M2 Parameter</b>	
SWComponentTemplate::PortInterface::InvalidationPolicy, SystemTemplate::Fibex::FibexCore::Core Communication::ISignalPort.handleInvalid	
<b>Mapping Rule</b>	<b>Mapping Type</b>
If strategy HandleInvalidEnum.keep is defined then set ComDataInvalidAction to NOTIFY. If strategy HandleInvalidEnum.replace is defined then set ComDataInvalidAction to REPLACE. In all other cases the ComDataInvalidAction shall not be configured.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Com_00314]

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComSignal	
<b>BSW Parameter</b>	<b>BSW Type</b>	
ComFilter	ECUC-PARAM-CONF-CONTAINER-DEF	
<b>BSW Description</b>		
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.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
Create container on the receiver side if the NonqueuedReceiverComSpec contains a DataFilter. Create Container on the sender side if the TransmissionModeCondition element contains a reference to this signal.	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Com_00339]	

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComSignal/ComFilter	
<b>BSW Parameter</b>	<b>BSW Type</b>	
ComFilterAlgorithm	ECUC-ENUMERATION-PARAM-DEF	
<b>BSW Description</b>		
The range of values is specified in the [17] specification, chapter 2.2.2, Reception Filtering.		
<b>Template Description</b>		







This attribute specifies the type of the filter.	
<b>M2 Parameter</b>	
CommonStructure::Filter::DataFilter.dataFilterType	
<b>Mapping Rule</b>	<b>Mapping Type</b>
Mapping between DataFilterTypeEnum and ComFilterAlgorithm Enum is necessary.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Com_00146]

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComSignal/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilterMask		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
<b>Template Description</b>		
Mask for old and new value.		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.mask		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Com_00235]	

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComSignal/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>
ComFilterMax		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
<b>Template Description</b>		
Value to specify the upper boundary		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.max		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Com_00317]	

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComSignal/ComFilter	
<b>BSW Parameter</b>		<b>BSW Type</b>





ComFilterMin	ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>	
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.	
<b>Template Description</b>	
Value to specify the lower boundary	
<b>M2 Parameter</b>	
CommonStructure::Filter::DataFilter.min	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Com_00318]

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComSignal/ComFilter	
<b>BSW Parameter</b>	<b>BSW Type</b>	
ComFilterOffset	ECUC-INTEGER-PARAM-DEF	
<b>BSW Description</b>		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering. Range = 0..(ComFilterPeriod-1)		
<b>Template Description</b>		
Specifies the initial number of messages to occur before the first message is passed		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.offset		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Com_00313]	

<b>BSW Module</b>	<b>BSW Context</b>	
Com	Com/ComConfig/ComSignal/ComFilter	
<b>BSW Parameter</b>	<b>BSW Type</b>	
ComFilterPeriod	ECUC-INTEGER-PARAM-DEF	
<b>BSW Description</b>		
This parameter defines the period of the ComFilterAlgorithm ONE_EVERY_N.		
<b>Template Description</b>		
Specifies number of messages to occur before the message is passed again		
<b>M2 Parameter</b>		
CommonStructure::Filter::DataFilter.period		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Com_00312]	

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal/ComFilter	
BSW Parameter	BSW Type	
ComFilterX	ECUC-INTEGER-PARAM-DEF	
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		ECUC Parameter ID
valid		[ECUC_Com_00147]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal	
BSW Parameter	BSW Type	
ComSignalDataInvalidValue	ECUC-STRING-PARAM-DEF	
BSW Description		
<p>Defines the data invalid value of the signal.</p> <p>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.</p>		
Template Description		
<p><b>InvalidationPolicy:</b> Specifies whether the component can actively invalidate a particular dataElement.</p> <p>If no invalidationPolicy points to a dataElement this is considered to yield the identical result as if the handleInvalid attribute was set to dontInvalidate.</p> <p><b>SwDataDefProps.invalidValue:</b> Optional value to express invalidity of the actual data element.</p>		
M2 Parameter		
SWComponentTemplate::PortInterface::InvalidationPolicy, DataDictionary::DataDefProperties::SwDataDefProps. <a href="#">invalidValue</a>		
Mapping Rule		Mapping Type
ComSignalDataInvalidValue is only derived 1:1 from the SwDataDefProps.invalidValue if the InvalidationPolicy equals keep or replace. In all other cases of InvalidationPolicy the ComSignalDataInvalidValue shall not be configured.		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Com_00391]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal	
BSW Parameter	BSW Type	
ComSignalInitValue	ECUC-STRING-PARAM-DEF	
BSW Description		
<p>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.</p> <p>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.</p>		
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. <a href="#">initValue</a> , SWComponentTemplate::Communication::NonqueuedSenderComSpec. <a href="#">initValue</a>		
Mapping Rule		Mapping Type
It is possible to aggregate an initValue at the level of a ComSpec in the SWC Template. in case the System Description doesn't use a complete Software Component Description (VFB View) the init Value is defined in the System Template.		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Com_00170]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal	
BSW Parameter	BSW Type	
ComTimeout	ECUC-FLOAT-PARAM-DEF	
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		
<p><b>ISignalPort.timeout:</b></p> <ul style="list-style-type: none"> <li>ISignalPort with communicationDirection = in:           <p>Optional timeout value in seconds for the reception of the ISignal. The attribute value is used to configure the ComTimeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the NonqueuedReceiverComSpec.aliveTimeout attribute. If a full DataMapping exists for the SystemSignal and the value is available in the configured ReceiverComSpec, then the timeout value in the ReceiverComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module.</p> </li> <li>ISignalPort with communicationDirection = out:           <p>Optional timeout value in seconds for the transmission of the ISignal. The attribute value is used to configure the ComTimeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the senderComSpec.transmissionAcknowledge.timeout attribute. If a full DataMapping exists for the SystemSignal and the value is available</p> </li> </ul>		





in the configured SenderComSpec, then the timeout value in the SenderComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module.

This attribute can be used in the following cases:

- legacy signal where the System Description doesn't use a complete Software Component Description (VFB View) and where the DataMapping is missing.
- bus monitoring use cases in which the DataMapping is ignored.

**TransmissionAcknowledgementRequest.timeout:**

Number of seconds before an error is reported or in case of allowed redundancy, the value is sent again.

**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.

**M2 Parameter**

SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignalPort.timeout, SWComponent  
Template::Communication::TransmissionAcknowledgementRequest.timeout, SWComponent  
Template::Communication::NonqueuedReceiverComSpec.aliveTimeout

**Mapping Rule**

TX Signals: If a full DataMapping exist for the SystemSignal this information may be available from a configured SenderComSpec.transmissionAcknowledge.timeout that specifies 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 transmitted according to the specified timing description. In this case the timeout value in SenderComSpec overrides the optional timeout specification in the System Template defined on the ISignalPort. RX Signals: If a full DataMapping exist for the SystemSignal this information may be available from a configured NonqueuedReceiverComSpec.aliveTimeout. In this case the timeout value in ReceiverComSpec overrides the optional timeout specification in the System Template defined on the ISignalPort. Please note that the SWS\_RTE defines an algorithm to finally set the applicable timeout value.

**Mapping Type**

full

**Mapping Status**

valid

**ECUC Parameter ID**

[ECUC\_Com\_00263]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal	
BSW Parameter	BSW Type	
ComTimeoutSubstitutionValue	ECUC-STRING-PARAM-DEF	
BSW Description		
<p>The signal substitution value will be used in case of a timeout and ComRxDataTimeoutAction is set to SUBSTITUTE. In case of UINT8_N the default value is a string of length ComSignalLength with all bytes set to 0x00.</p> <p>In case ofUINT8_DYN the initial size shall be 0.</p> <p>In case the ComSignalType is UINT8, UINT16, UIN32, UIN64, SINT8, SINT16, SINT32, SINT64 the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification.</p> <p>In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification.</p> <p>In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification.</p> <p>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.</p>		
Template Description		
Defines and enables the ComTimeoutSubstitution for this ISignal.		





<b>M2 Parameter</b>	
SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal.timeoutSubstitutionValue	
<b>Mapping Rule</b>	<b>Mapping Type</b>
The mapping of ComTimeoutSubstitutionValue depends on the setting in the ISignal.dataTypePolicy: - ISignal.dataTypePolicy = override or legacy: SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal.timeoutSubstitutionValue - ISignal.dataTypePolicy = networkRepresentationFromComSpec: SWComponentTemplate::Communication::NonequeuedReceiverComSpec.timeoutSubstitutionValue - ISignal.dataTypePolicy = transformingISignal this is not supported.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Com_10006]

<b>BSW Module</b>	<b>BSW Context</b>
Com	Com/ComConfig/ComSignalGroup
<b>BSW Parameter</b>	<b>BSW Type</b>
ComDataInvalidAction	ECUC-ENUMERATION-PARAM-DEF
<b>BSW Description</b>	
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.	
<b>Template Description</b>	
<p><b>InvalidationPolicy:</b> Specifies whether the component can actively invalidate a particular dataElement.</p> <p>If no invalidationPolicy points to a dataElement this is considered to yield the identical result as if the handleInvalid attribute was set to dontInvalidate.</p> <p><b>ISignalPort.handleInvalid:</b> This attribute defines how invalidation is applied to the ISignals received in the context of this ISignalPort.</p>	
<b>M2 Parameter</b>	
SWComponentTemplate::PortInterface::InvalidationPolicy, SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignalPort.handleInvalid	
<b>Mapping Rule</b>	<b>Mapping Type</b>
If strategy HandleInvalidEnum.keep is defined then set ComDataInvalidAction to NOTIFY. If strategy HandleInvalidEnum.replace is defined then set ComDataInvalidAction to REPLACE. In all other cases the ComDataInvalidAction shall not be configured.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Com_00314]

<b>BSW Module</b>	<b>BSW Context</b>
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal
<b>BSW Parameter</b>	<b>BSW Type</b>
ComFilter	ECUC-PARAM-CONF-CONTAINER-DEF
<b>BSW Description</b>	
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.	
<b>Template Description</b>	
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 TransmissionModeCondition element contains a reference to this signal.	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Com_00339]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterAlgorithm		ECUC-ENUMERATION-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_Com_00146]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterMask		ECUC-INTEGER-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_Com_00235]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	





BSW Parameter		BSW Type
ComFilterMax		ECUC-INTEGER-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_Com_00317]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterMin		ECUC-INTEGER-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_Com_00318]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterOffset		ECUC-INTEGER-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_Com_00313]



BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterPeriod		ECUC-INTEGER-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_Com_00312]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterX		ECUC-INTEGER-PARAM-DEF
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		ECUC Parameter ID
valid		[ECUC_Com_00147]

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal	
BSW Parameter		BSW Type
ComSignalDataInvalidValue		ECUC-STRING-PARAM-DEF
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.	
<b>Template Description</b>	
<b>InvalidationPolicy:</b> 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.  <b>SwDataDefProps.invalidValue:</b> Optional value to express invalidity of the actual data element.	
<b>M2 Parameter</b>	
SWComponentTemplate::PortInterface::InvalidationPolicy, DataDictionary::DataDefProperties::SwDataDefProps. <a href="#">invalidValue</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
ComSignalDataInvalidValue is only derived 1:1 from the SwDataDefProps.invalidValue if the InvalidationPolicy equals keep or replace. In all other cases of InvalidationPolicy the ComSignalDataInvalidValue shall not be configured.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Com_00391]

<b>BSW Module</b>	<b>BSW Context</b>
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal
<b>BSW Parameter</b>	<b>BSW Type</b>
ComSignalInitValue	ECUC-STRING-PARAM-DEF
<b>BSW Description</b>	
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.	
<b>Template Description</b>	
Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.	
<b>M2 Parameter</b>	
SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal. <a href="#">initValue</a> , SWComponentTemplate::Communication::NonqueuedSenderComSpec. <a href="#">initValue</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
It is possible to aggregate an initValue at the level of a ComSpec in the SWC Template. in case the System Description doesn't use a complete Software Component Description (VFB View) the init Value is defined in the System Template.	full





Mapping Status	ECUC Parameter ID
valid	[ECUC_Com_00170]

BSW Module	BSW Context
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal
BSW Parameter	BSW Type
ComTimeoutSubstitutionValue	ECUC-STRING-PARAM-DEF
BSW Description	
<p>The signal substitution value will be used in case of a timeout and ComRxDataTimeoutAction is set to SUBSTITUTE. In case of UINT8_N the default value is a string of length ComSignalLength with all bytes set to 0x00.</p> <p>In case of UINT8_DYN the initial size shall be 0.</p> <p>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.</p> <p>In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification.</p> <p>In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification.</p> <p>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.</p>	
Template Description	
Defines and enables the ComTimeoutSubstitution for this ISignal.	
M2 Parameter	
SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal.timeoutSubstitutionValue	
Mapping Rule	Mapping Type
The mapping of ComTimeoutSubstitutionValue depends on the setting in the ISignal.dataTypePolicy: - ISignal.dataTypePolicy = override or legacy: SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal.timeoutSubstitutionValue - ISignal.dataTypePolicy = networkRepresentationFromComSpec: SWComponentTemplate::Communication::NonequeuedReceiverComSpec.timeoutSubstitutionValue - ISignal.dataTypePolicy = transformIngISignal this is not supported.	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Com_10006]

BSW Module	BSW Context
Com	Com/ComConfig/ComSignalGroup
BSW Parameter	BSW Type
ComTimeout	ECUC-FLOAT-PARAM-DEF
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	





<p><b>ISignalPort.timeout:</b></p> <ul style="list-style-type: none"> <li>ISignalPort with communicationDirection = in: Optional timeout value in seconds for the reception of the ISignal. The attribute value is used to configure the ComTimeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the NonqueuedReceiverComSpec.aliveTimeout attribute. If a full DataMapping exists for the SystemSignal and the value is available in the configured ReceiverComSpec, then the timeout value in the ReceiverComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module.</li> <li>ISignalPort with communicationDirection = out: Optional timeout value in seconds for the transmission of the ISignal. The attribute value is used to configure the Com Timeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the enderComSpec.transmissionAcknowledge.timeout attribute. If a full DataMapping exists for the SystemSignal and the value is available in the configured SenderComSpec, then the timeout value in the SenderComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module.</li> </ul> <p>This attribute can be used in the following cases:</p> <ul style="list-style-type: none"> <li>legacy signal where the System Description doesn't use a complete Software Component Description (VFB View) and where the DataMapping is missing.</li> <li>bus monitoring use cases in which the DataMapping is ignored.</li> </ul> <p><b>TransmissionAcknowledgementRequest.timeout:</b> Number of seconds before an error is reported or in case of allowed redundancy, the value is sent again.</p> <p><b>NonqueuedReceiverComSpec.aliveTimeout:</b> 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.</p> <p>If the aliveTimeout attribute is 0 no timeout monitoring shall be performed.</p>	
<p><b>M2 Parameter</b></p> <p>SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignalPort.timeout, SWComponent                  Template::Communication::TransmissionAcknowledgementRequest.timeout, SWComponent                  Template::Communication::NonqueuedReceiverComSpec.aliveTimeout</p>	
<p><b>Mapping Rule</b></p> <p>TX Signals: If a full DataMapping exist for the SystemSignal this information may be available from a configured SenderComSpec.transmissionAcknowledge.timeout that specifies 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 transmitted according to the specified timing description. In this case the timeout value in SenderComSpec overrides the optional timeout specification in the System Template defined on the ISignalPort. RX Signals: If a full DataMapping exist for the SystemSignal this information may be available from a configured NonqueuedReceiverComSpec.aliveTimeout. In this case the timeout value in ReceiverComSpec overrides the optional timeout specification in the System Template defined on the ISignalPort. Please note that the SWS_RTE defines an algorithm to finally set the applicable timeout value.</p>	<p><b>Mapping Type</b></p> <p>full</p>
<p><b>Mapping Status</b></p> <p>valid</p>	<p><b>ECUC Parameter ID</b></p> <p>[ECUC_Com_00263]</p>

## G.4 WdgM

<b>BSW Module</b>	<b>BSW Context</b>
WdgM	WdgM/WdgMConfigSet/WdgMMode/WdgMLocalStatusParams
<b>BSW Parameter</b>	<b>BSW Type</b>
WdgMFailedAliveSupervisionRefCycleTol	ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>	
This parameter shall contain the acceptable amount of reference cycles with incorrect/failed alive supervisions for this Supervised Entity.	





Template Description	
<p>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).</p> <p>Note that this value has to be recalculated with respect to the Wdgm's own cycle time for ECU configuration.</p>	
M2 Parameter	
CommonStructure::ServiceNeeds::SupervisedEntityNeeds.toleratedFailedCycles	
Mapping Rule	Mapping Type
1:1	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Wdgm_00327]

## G.5 Dcm

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsd	
BSW Parameter		BSW Type
DcmDsdServiceRequestManufacturerNotification		ECUC-PARAM-CONF-CONTAINER-DEF
BSW Description		
<p>The name of this container is used to define the name of the R-Port through which the DCM accesses the interface ServiceRequestNotification. The R-Port is named ServiceRequestManufacturerNotification_{Name} where {Name} is the name of the container DcmDsdServiceRequestManufacturerNotification.</p> <p>The lowerMultiplicity is 0: If container DcmDsdServiceRequestManufacturerNotification does not exist the Indication API is not available.</p>		
Template Description		
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.		
M2 Parameter		
CommonStructure::ServiceNeeds::DiagnosticCommunicationManagerNeeds.serviceRequestCallbackType		
Mapping Rule		Mapping Type
If DiagnosticCommunicationManagerNeeds.serviceRequestCallbackType is set to DiagnosticServiceRequestCallbackTypeEnum.requestCallbackTypeManufacturer then DcmDsdServiceRequestManufacturerNotification shall exist and the value of DcmDsdServiceRequestManufacturerNotification.shortName shall be taken from the SwcServiceDependency.shortName that aggregates the DiagnosticCommunicationManagerNeeds.		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_00681]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsd	
BSW Parameter		BSW Type
DcmDsdServiceRequestSupplierNotification		ECUC-PARAM-CONF-CONTAINER-DEF
BSW Description		





<p>The name of this container is used to define the name of the R-Port through which the DCM accesses the interface ServiceRequestNotification. The R-Port is named ServiceRequestSupplierNotification_&lt;SWC&gt; where &lt;SWC&gt; is the name of the container DcmDsdServiceRequestSupplierNotification.</p> <p>The lowerMultiplicity is 0: If the container DcmDsdRequestSupplierNotification does not exist the Indication API is not available.</p>	
<b>Template Description</b>	
<p>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.</p>	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::DiagnosticCommunicationManagerNeeds.serviceRequestCallbackType	
<b>Mapping Rule</b>	<b>Mapping Type</b>
<p>If DiagnosticCommunicationManagerNeeds.serviceRequestCallbackType is set to DiagnosticServiceRequestCallbackTypeEnum.requestCallbackTypeSupplier then DcmDsdServiceRequestSupplierNotification shall exist and the value of DcmDsdServiceRequestSupplierNotification.shortName shall be taken from the SwcServiceDependency.shortName that aggregates the DiagnosticCommunicationManagerNeeds.</p>	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_00816]

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData
<b>BSW Parameter</b>	<b>BSW Type</b>
DcmDspDataResetToDefaultFnc	ECUC-FUNCTION-NAME-DEF
<b>BSW Description</b>	
<p>Function name to request to application to reset an IOControl to default value. (ResetToDefault-function). This parameter is related to the interface Xxx_ResetToDefault.</p>	
<b>Template Description</b>	
<p><b>DiagnosticIoControlNeeds.resetToDefaultSupported:</b> This represents a flag for the existence of the ResetToDefault operation in the service interface.</p> <p><b>DiagnosticServiceSwMapping.mappedBswServiceDependency:</b> This is supposed to represent a reference to a BswServiceDependency. the latter is not derived from Referrable and therefore this detour needs to be implemented to still let BswServiceDependency become the target of a reference.</p>	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::DiagnosticIoControlNeeds.resetToDefaultSupported, DiagnosticExtract::DiagnosticMapping::ServiceMapping::DiagnosticServiceSwMapping.mappedBswServiceDependency	
<b>Mapping Rule</b>	<b>Mapping Type</b>
It could be possible to get the FNC name via BswServiceDependency	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_00673]

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData
<b>BSW Parameter</b>	<b>BSW Type</b>
DcmDspDataShortTermAdjustmentFnc	ECUC-FUNCTION-NAME-DEF
<b>BSW Description</b>	





Function name to request to application to adjust the IO signal. (ShortTermAdjustment-function).	
This parameter is related to the interface Xxx_ShortTermAdjustment.	
<b>Template Description</b>	
<b>DiagnosticIoControlNeeds.shortTermAdjustmentSupported:</b> This attribute determines, if the referenced port supports temporarily setting of I/O value to a specific value provided by the diagnostic tester.	
<b>DiagnosticServiceSwMapping.mappedBswServiceDependency:</b> This is supposed to represent a reference to a BswServiceDependency. the latter is not derived from Referrable and therefore this detour needs to be implemented to still let BswServiceDependency become the target of a reference.	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::DiagnosticIoControlNeeds.shortTermAdjustmentSupported, DiagnosticExtract::DiagnosticMapping::ServiceMapping::DiagnosticServiceSwMapping.mappedBswServiceDependency	
<b>Mapping Rule</b>	<b>Mapping Type</b>
It could be possible to get the FNC name via BswServiceDependency	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_00675]

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType	
<b>BSW Parameter</b>	<b>BSW Type</b>	
BOOLEAN	ECUC-ENUMERATION-LITERAL-DEF	
<b>BSW Description</b>		
Type of the data is boolean.		
<b>Template Description</b>		
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.		
<b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.		
<b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency.  This attribute controls whether the data length of the data is fixed.		
<b>M2 Parameter</b>		
AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeEncoding, AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeSize, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.fixedLength		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
baseTypeEncoding = BOOLEAN baseTypeSize = 1 maxNumberOfElements shall not exist array SizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid		

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType	
<b>BSW Parameter</b>	<b>BSW Type</b>	
FLOAT	ECUC-ENUMERATION-LITERAL-DEF	
<b>BSW Description</b>		





Type of the data is float.	
<b>Template Description</b>	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
<b>BSW Parameter</b>	<b>BSW Type</b>
FLOAT_N	ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>	
Type of the data is float array.	
<b>Template Description</b>	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
<b>BSW Parameter</b>	<b>BSW Type</b>







SINT16	ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>	
Type of the data is sint16.	
<b>Template Description</b>	
<p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = 2C baseTypeSize = 16 maxNumberOfElements shall not exist arraySize Semantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType	
<b>BSW Parameter</b>		<b>BSW Type</b>
SINT16_N		ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>		
Type of the data is sint16 array.		
<b>Template Description</b>		
<p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>DiagnosticDataElement.arraySizeSemantics:</b> This attribute controls the meaning of the value of the array size.</p> <p><b>DiagnosticDataElement.maxNumberOfElements:</b> 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 how many elements the array can take.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>		
<b>M2 Parameter</b>		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">arraySizeSemantics</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">maxNumberOfElements</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
<b>Mapping Rule</b>		<b>Mapping Type</b>
baseTypeEncoding = 2C baseTypeSize = 16 maxNumberOfElements exists and value is greater than 0 (cf. TPS_DEXT_01001) arraySizeSemantics either does not exist or exists and is set to ArraySizeSemanticsEnum.fixedSize (cf. TPS_DEXT_01001) Derivation from DiagnosticValueNeeds.fixedLength=1 possible.		full





Mapping Status	ECUC Parameter ID
valid	

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
BSW Parameter	BSW Type
SINT32	ECUC-ENUMERATION-LITERAL-DEF
BSW Description	
Type of the data is sint32.	
Template Description	
<p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
M2 Parameter	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
Mapping Rule	Mapping Type
baseTypeEncoding = 2C baseTypeSize = 32 maxNumberOfElements shall not exist arraySize Semantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.	full
Mapping Status	ECUC Parameter ID
valid	

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
BSW Parameter	BSW Type
SINT32_N	ECUC-ENUMERATION-LITERAL-DEF
BSW Description	
Type of the data is sint32 array.	
Template Description	
<p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>DiagnosticDataElement.arraySizeSemantics:</b> This attribute controls the meaning of the value of the array size.</p> <p><b>DiagnosticDataElement.maxNumberOfElements:</b> 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 how many elements the array can take.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	





<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">arraySizeSemantics</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">maxNumberOfElements</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = 2C baseTypeSize = 32 maxNumberOfElements exists and value is greater than 0 (cf. TPS_DEXT_01001) arraySizeSemantics either does not exist or exists and is set to ArraySizeSemanticsEnum.fixedSize (cf. TPS_DEXT_01001) Derivation from DiagnosticValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
<b>BSW Parameter</b>	<b>BSW Type</b>
SINT8	ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>	
Type of the data is sint8.	
<b>Template Description</b>	
<b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.	
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.	
<b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = 2C baseTypeSize = 8 maxNumberOfElements shall not exist arraySize Semantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
<b>BSW Parameter</b>	<b>BSW Type</b>
SINT8_N	ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>	
Type of the data is sint8 array.	
<b>Template Description</b>	





<p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>DiagnosticDataElement.arraySizeSemantics:</b> This attribute controls the meaning of the value of the array size.</p> <p><b>DiagnosticDataElement.maxNumberOfElements:</b> 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 how many elements the array can take.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
<b>M2 Parameter</b>	
<p>AsamHdo::BaseTypes::BaseTypeDirectDefinition.<a href="#">baseTypeSize</a>, AsamHdo::BaseTypes::BaseTypeDirectDefinition.<a href="#">baseTypeEncoding</a>, DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement.<a href="#">arraySizeSemantics</a>, DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement.<a href="#">maxNumberOfElements</a>, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.<a href="#">fixedLength</a></p>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
<p>baseTypeEncoding = 2C baseTypeSize = 8 maxNumberOfElements exists and value is greater than 0 (cf. TPS_DEXT_01001) arraySizeSemantics either does not exist or exists and is set to ArraySizeSemanticsEnum.fixedSize (cf. TPS_DEXT_01001) Derivation from DiagnosticValueNeeds.fixedLength=1 possible.</p>	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
<b>BSW Parameter</b>	<b>BSW Type</b>
UINT16	ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>	
Type of the data is uint16.	
<b>Template Description</b>	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
<b>M2 Parameter</b>	
<p>AsamHdo::BaseTypes::BaseTypeDirectDefinition.<a href="#">baseTypeEncoding</a>, AsamHdo::BaseTypes::BaseTypeDirectDefinition.<a href="#">baseTypeSize</a>, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.<a href="#">fixedLength</a></p>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
<p>baseTypeEncoding = NONE, UTF-16 baseTypeSize = 16 maxNumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.</p>	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType	
BSW Parameter	BSW Type	
UINT16_N	ECUC-ENUMERATION-LITERAL-DEF	
BSW Description		
Type of the data is uint16 array.		
Template Description		
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticDataElement.arraySizeSemantics:</b> This attribute controls the meaning of the value of the array size.</p> <p><b>DiagnosticDataElement.maxNumberOfElements:</b> 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 how many elements the array can take.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">arraySizeSemantics</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">maxNumberOfElements</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
Mapping Rule		Mapping Type
baseTypeEncoding = NONE, UTF-16 baseTypeSize = 16 maxNumberOfElements exists and value is greater than 0 (cf. TPS_DEXT_01001) arraySizeSemantics either does not exist or exists and is set to ArraySizeSemanticsEnum.fixedSize (cf. TPS_DEXT_01001) Derivation from DiagnosticValueNeeds.fixedLength=1 possible.		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType	
BSW Parameter	BSW Type	
UINT32	ECUC-ENUMERATION-LITERAL-DEF	
BSW Description		
Type of the data is uint32.		
Template Description		
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>		
M2 Parameter		





AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = NONE, UTF-32 baseTypeSize = 32 maxNumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
<b>BSW Parameter</b>	<b>BSW Type</b>
UINT32_N	ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>	
Type of the data is uint32 array.	
<b>Template Description</b>	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticDataElement.arraySizeSemantics:</b> This attribute controls the meaning of the value of the array size.</p> <p><b>DiagnosticDataElement.maxNumberOfElements:</b> 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 how many elements the array can take.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">arraySizeSemantics</a> , Diagnostic Extract::CommonDiagnostics::DiagnosticDataElement. <a href="#">maxNumberOfElements</a> , CommonStructure::Service Needs::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = NONE, UTF-32 baseTypeSize = 32 maxNumberOfElements exists and value is greater than 0 (cf. TPS_DEXT_01001) arraySizeSemantics either does not exist or exists and is set to ArraySizeSemanticsEnum.fixedSize (cf. TPS_DEXT_01001) Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
<b>BSW Parameter</b>	<b>BSW Type</b>
UINT8	ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>	
Type of the data is uint8.	





Template Description	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
M2 Parameter	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
Mapping Rule	Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
Mapping Status	ECUC Parameter ID
valid	

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType
BSW Parameter	BSW Type
UINT8_DYN	ECUC-ENUMERATION-LITERAL-DEF
BSW Description	
Type of the data is uint8 array with dynamic length.	
Template Description	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticDataElement.arraySizeSemantics:</b> This attribute controls the meaning of the value of the array size.</p> <p><b>DiagnosticDataElement.maxNumberOfElements:</b> 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 how many elements the array can take.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
M2 Parameter	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">arraySizeSemantics</a> , Diagnostic Extract::CommonDiagnostics::DiagnosticDataElement. <a href="#">maxNumberOfElements</a> , CommonStructure::Service Needs::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
Mapping Rule	Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements exists and value is greater than 0 (cf. TPS_DEXT_01002) arraySizeSemantics exists and is set to ArraySizeSemanticsEnum.variableSize (cf. TPS_DEXT_01002) Derivation from DiagnosticValueNeeds.fixedLength=0 possible.	full
Mapping Status	ECUC Parameter ID
valid	

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataType	
BSW Parameter	BSW Type	
UINT8_N	ECUC-ENUMERATION-LITERAL-DEF	
BSW Description		
Type of the data is uint8 array.		
Template Description		
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticDataElement.arraySizeSemantics:</b> This attribute controls the meaning of the value of the array size.</p> <p><b>DiagnosticDataElement.maxNumberOfElements:</b> 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 how many elements the array can take.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">arraySizeSemantics</a> , DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement. <a href="#">maxNumberOfElements</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
Mapping Rule		Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements exists and value is greater than 0 (cf. TPS_DEXT_01001) arraySizeSemantics either does not exist or exists and is set to ArraySizeSemanticsEnum.fixedSize (cf. TPS_DEXT_01001) Derivation from DiagnosticValueNeeds.fixedLength=1 possible.		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData	
BSW Parameter	BSW Type	
DcmDspDataUsePort	ECUC-ENUMERATION-PARAM-DEF	
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. <a href="#">processingStyle</a>		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_00713]



BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort	
BSW Parameter	BSW Type	
USE_DATA_ASYNCH_CLIENT_SERVER	ECUC-ENUMERATION-LITERAL-DEF	
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. <a href="#">processingStyleAsynchronous</a>		
Mapping Rule		Mapping Type
DiagnosticServiceSwMapping is having a SwcServiceDependency and ServiceNeeds::Diagnostic ProcessingStyleEnum is equal to processingStyleAsynchronous		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort	
BSW Parameter	BSW Type	
USE_DATA_ASYNCH_CLIENT_SERVER_ERROR	ECUC-ENUMERATION-LITERAL-DEF	
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 processes the request in background but still the Dcm has to issue the call again to eventually obtain the result of the request or handle error code.		
M2 Parameter		
CommonStructure::ServiceNeeds::DiagnosticProcessingStyleEnum. <a href="#">processingStyleAsynchronousWithError</a>		
Mapping Rule		Mapping Type
DiagnosticServiceSwMapping is having a SwcServiceDependency and ServiceNeeds::Diagnostic ProcessingStyleEnum is equal to processingStyleAsynchronousWithError		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort	
BSW Parameter	BSW Type	
USE_DATA_SYNCH_CLIENT_SERVER	ECUC-ENUMERATION-LITERAL-DEF	
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.	
<b>Template Description</b>	
The software-component is supposed to react synchronously on the request.	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::DiagnosticProcessingStyleEnum.processingStyleSynchronous	
<b>Mapping Rule</b>	<b>Mapping Type</b>
DiagnosticServiceSwMapping is having a SwcServiceDependency and ServiceNeeds::DiagnosticProcessingStyleEnum is equal to processingStyleSynchronous	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDiagnosisScaling/DcmDspAlternativeData Interface	
<b>BSW Parameter</b>		<b>BSW Type</b>
DcmDataElement		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Alternative Diagnosis Representation for the data defined by the means of a VariableDataPrototype in a DataInterface. The CompuMethod of the data type of the referenced VariableDataPrototype will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
SWComponentTemplate::Datatype::DataPrototypes::VariableDataPrototype		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Dcm_00995]

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDiagnosisScaling/DcmDspAlternativeData Interface	
<b>BSW Parameter</b>		<b>BSW Type</b>
DcmPortInterfaceMapping		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Optional reference to PortInterfaceMapping which defines the mapping rules.  The PortInterfaceMapping is used to get the DataPrototypeMapping that describes a conversion between the data prototype referenced by DcmDataElement and the data prototype referenced from DcmDspExternalSRDataElementClass.		
<b>Template Description</b>		
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).		





<b>M2 Parameter</b>	
SWComponentTemplate::PortInterface::PortInterfaceMapping	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_00996]

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDiagnosisScaling/DcmDspAlternativeDataType
<b>BSW Parameter</b>	<b>BSW Type</b>
DcmApplicationDataType	ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>	
Alternative Diagnosis Representation for the data defined by the means of a ApplicationDataType of category VALUE, BOOLEAN or ARRAY. The CompuMethod that applies to the referenced ApplicationDataType in case of category VALUE or BOOLEAN will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.	
<b>Template Description</b>	
A primitive data type defines a set of allowed values.	
<b>M2 Parameter</b>	
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_00998]

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl
<b>BSW Parameter</b>	<b>BSW Type</b>
DcmDspDidFreezeCurrentState	ECUC-BOOLEAN-PARAM-DEF
<b>BSW Description</b>	
This indicates the presence of "FreezeCurrentState".	
<b>Template Description</b>	
<b>DiagnosticIOControl.freezeCurrentState:</b> Setting this attribute to true represents the ability of the Dcm to execute a freezeCurrentState.	
<b>DiagnosticControlNeeds.freezeCurrentStateSupported:</b> This attribute determines, if the referenced port supports temporary freezing of I/O value.	
<b>M2 Parameter</b>	
DiagnosticExtract::Dcm::DiagnosticService::IOControl::DiagnosticIOControl.freezeCurrentState, Common Structure::ServiceNeeds::DiagnosticControlNeeds.freezeCurrentStateSupported	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_00624]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl	
BSW Parameter	BSW Type	
DcmDspDidResetToDefault	ECUC-BOOLEAN-PARAM-DEF	
BSW Description		
This indicates the presence of "ResetToDefault".		
Template Description		
<b>DiagnosticIOControl.resetToDefault:</b> Setting this attribute to true represents the ability of the Dcm to execute a resetToDefault.		
<b>DiagnosticIOControlNeeds.resetToDefaultSupported:</b> 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		ECUC Parameter ID
valid		[ECUC_Dcm_00623]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl	
BSW Parameter	BSW Type	
DcmDspDidShortTermAdjustment	ECUC-BOOLEAN-PARAM-DEF	
BSW Description		
This indicates the presence of "ShortTermAdjustment".		
Template Description		
<b>DiagnosticIOControl.shortTermAdjustment:</b> Setting this attribute to true represents the ability of the Dcm to execute a shortTermAdjustment.		
<b>DiagnosticIOControlNeeds.shortTermAdjustmentSupported:</b> 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		ECUC Parameter ID
valid		[ECUC_Dcm_00625]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspMemoryTransfer	
BSW Parameter	BSW Type	
DcmDspMemoryTransferUsePort	ECUC-BOOLEAN-PARAM-DEF	





<b>BSW Description</b>	
If this parameter is set to true, the Dcm uses a port requiring a PortInterface UploadDownload. If the parameter is false, the DCM uses the according C-API callouts.	
<b>Template Description</b>	
This meta-class represents the ability to specify needs regarding upload and download by means of diagnostic services.	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::DiagnosticUploadDownloadNeeds	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_01133]

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData	
<b>BSW Parameter</b>		<b>BSW Type</b>
DcmDspPidDataByteSize		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
Defines the array length in bytes or the the maximum array length for variable datalengths.		
<b>Template Description</b>		
<b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.		
<b>DiagnosticDataElement.maxNumberOfElements:</b> 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 how many elements the array can take.		
<b>M2 Parameter</b>		
AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeSize, DiagnosticExtract::CommonDiagnostics::DiagnosticDataElement.maxNumberOfElements		
<b>Mapping Rule</b>		<b>Mapping Type</b>
S/R via array: DcmDspPidDataByteSize= maxNumberOfElements * (baseTypeSize / 8) C/S of FNC callback: DcmDspPidDataByteSize= maxNumberOfElements Note: 8 is the baseTypeSize of UINT8		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Dcm_01108]

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspDiagnosisScaling/DcmDspAlternativeDataInterface	
<b>BSW Parameter</b>		<b>BSW Type</b>
DcmDataElement		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Alternative Diagnosis Representation for the data defined by the means of a VariableDataPrototype in a DataInterface. The CompuMethod of the data type of the referenced VariableDataPrototype will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.		
<b>Template Description</b>		





<p>A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableData Prototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided.</p> <p>In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.</p>	
<b>M2 Parameter</b>	
SWComponentTemplate::Datatype::DataPrototypes::VariableDataPrototype	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_00995]

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDsp DiagnosisScaling/DcmDspAlternativeDataInterface
<b>BSW Parameter</b>	<b>BSW Type</b>
DcmPortInterfaceMapping	ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>	
<p>Optional reference to PortInterfaceMapping which defines the mapping rules.</p> <p>The PortInterfaceMapping is used to get the DataPrototypeMapping that describes a conversion between the data prototype referenced by DcmDataElement and the data prototype referenced from DcmDspExternalSRDataElementClass.</p>	
<b>Template Description</b>	
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).	
<b>M2 Parameter</b>	
SWComponentTemplate::PortInterface::PortInterfaceMapping	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_00996]

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDsp DiagnosisScaling/DcmDspAlternativeDataType
<b>BSW Parameter</b>	<b>BSW Type</b>
DcmApplicationDataType	ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>	
<p>Alternative Diagnosis Representation for the data defined by the means of a ApplicationDataType of category VALUE, BOOLEAN or ARRAY.</p> <p>The CompuMethod that applies to the referenced ApplicationDataType in case of category VALUE or BOOLEAN will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.</p>	
<b>Template Description</b>	
A primitive data type defines a set of allowed values.	
<b>M2 Parameter</b>	
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType	





Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Dcm_00998]

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine
BSW Parameter	BSW Type
DcmDspRequestRoutineResults	ECUC-PARAM-CONF-CONTAINER-DEF
BSW Description	
Provides the configuration of RequestResult subservice for RoutineControl service. Existence indicates that the Request RoutineResults in the RoutineControl is supported.	
Template Description	
<b>DiagnosticRoutine.requestResult:</b> This represents the ability to request the result of a running routine.	
<b>DiagnosticRoutineNeeds.diagRoutineType:</b> This denotes the type of diagnostic routine which is implemented by the referenced server port.	
M2 Parameter	
DiagnosticExtract::CommonDiagnostics::DiagnosticRoutine.requestResult, CommonStructure::ServiceNeeds::Diagnostic RoutineNeeds.diagRoutineType	
Mapping Rule	Mapping Type
1:1 mapping for DiagnosticRoutine.requestResult OR DiagnosticRoutineNeeds.diagRoutineType == asynchronous	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Dcm_01023]

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequest RoutineResultsIn/DcmDspRequestRoutineResultsInSignal/DcmDspArgumentScaling/DcmDsp AlternativeArgumentData
BSW Parameter	BSW Type
DcmDataElement	ECUC-FOREIGN-REFERENCE-DEF
BSW Description	
Alternative Diagnosis Representation for the data defined by the means of a ArgumentDataPrototype. The CompuMethod of the data type of the referenced ArgumentDataPrototype will be applied to the data type of the ArgumentDataPrototype in the interface used by the Dcm.	
Template Description	
An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.	
M2 Parameter	
SWComponentTemplate::PortInterface::ArgumentDataPrototype	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Dcm_01056]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal/DcmDspArgumentScaling/DcmDspAlternativeDataType	
BSW Parameter		BSW Type
DcmApplicationDataType		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Alternative Diagnosis Representation for the data defined by the means of a ApplicationDataType of category VALUE, BOOLEAN or ARRAY. The CompuMethod that applies to the referenced ApplicationDataType in case of category VALUE or BOOLEAN will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.		
Template Description		
A primitive data type defines a set of allowed values.		
M2 Parameter		
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_00998]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal/DcmDspRoutineSignalType	
BSW Parameter		BSW Type
FLOAT		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Type of the data is float.		
Template Description		
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.  <b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.  <b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeEncoding, AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeSize, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.fixedLength		
Mapping Rule		Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.		full
Mapping Status		ECUC Parameter ID
valid		



BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal/DcmDspRoutineSignalType	
BSW Parameter		BSW Type
FLOAT_N		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Type of the data is float array.		
Template Description		
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
Mapping Rule		Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal/DcmDspArgumentScaling/DcmDspAlternativeArgumentData	
BSW Parameter		BSW Type
DcmDataElement		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Alternative Diagnosis Representation for the data defined by the means of a ArgumentDataPrototype. The CompuMethod of the data type of the referenced ArgumentDataPrototype will be applied to the data type of the ArgumentDataPrototype in the interface used by the Dcm.		
Template Description		
An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.		
M2 Parameter		
SWComponentTemplate::PortInterface::ArgumentDataPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_01056]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal/DcmDspArgumentScaling/DcmDspAlternativeDataType	
BSW Parameter		BSW Type
DcmApplicationDataType		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Alternative Diagnosis Representation for the data defined by the means of a ApplicationDataType of category VALUE, BOOLEAN or ARRAY. The CompuMethod that applies to the referenced ApplicationDataType in case of category VALUE or BOOLEAN will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.		
Template Description		
A primitive data type defines a set of allowed values.		
M2 Parameter		
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_00998]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal/DcmDspRoutineSignalType	
BSW Parameter		BSW Type
FLOAT		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Type of the data is float.		
Template Description		
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.  <b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.  <b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeEncoding, AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeSize, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.fixedLength		
Mapping Rule		Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal/DcmDspRoutineSignalType	
BSW Parameter		BSW Type
FLOAT_N		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Type of the data is float array.		
Template Description		
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
Mapping Rule		Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine	
BSW Parameter		BSW Type
DcmDspStartRoutine		ECUC-PARAM-CONF-CONTAINER-DEF
BSW Description		
Provides the configuration of Start subservice for RoutineControl service.		
Template Description		
<p><b>DiagnosticRoutine.start:</b> This represents the ability to start a routine</p> <p><b>DiagnosticRoutineNeeds:</b> 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.</p>		
M2 Parameter		
DiagnosticExtract::CommonDiagnostics::DiagnosticRoutine. <a href="#">start</a> , CommonStructure::ServiceNeeds:: <a href="#">DiagnosticRoutineNeeds</a>		
Mapping Rule		Mapping Type
A routine always comes with a start routine, independently of whether the execution is done synchronously or asynchronously.		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_01021]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal/DcmDspArgumentScaling/DcmDspAlternativeArgumentData	
BSW Parameter		BSW Type
DcmDataElement		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
<p>Alternative Diagnosis Representation for the data defined by the means of a <code>ArgumentDataPrototype</code>.</p> <p>The <code>CompuMethod</code> of the data type of the referenced <code>ArgumentDataPrototype</code> will be applied to the data type of the <code>ArgumentDataPrototype</code> in the interface used by the <code>Dcm</code>.</p>		
Template Description		
An argument of an operation, much like a data element, but also carries direction information and is owned by a particular <code>ClientServerOperation</code> .		
M2 Parameter		
SWComponentTemplate::PortInterface::ArgumentDataPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_01056]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal/DcmDspArgumentScaling/DcmDspAlternativeDataType	
BSW Parameter		BSW Type
DcmApplicationDataType		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
<p>Alternative Diagnosis Representation for the data defined by the means of a <code>ApplicationDataType</code> of category <code>VALUE</code>, <code>BOOLEAN</code> or <code>ARRAY</code>.</p> <p>The <code>CompuMethod</code> that applies to the referenced <code>ApplicationDataType</code> in case of category <code>VALUE</code> or <code>BOOLEAN</code> will be applied to the data type of the <code>VariableDataPrototype</code> in the interface used by the <code>Dcm</code>.</p>		
Template Description		
A primitive data type defines a set of allowed values.		
M2 Parameter		
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_00998]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal/DcmDspRoutineSignalType	
BSW Parameter		BSW Type
FLOAT		ECUC-ENUMERATION-LITERAL-DEF





<b>BSW Description</b>	
Type of the data is float.	
<b>Template Description</b>	
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.	
<b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.	
<b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency.  This attribute controls whether the data length of the data is fixed.	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/Dcm DspStartRoutineInSignal/DcmDspRoutineSignalType	
<b>BSW Parameter</b>	<b>BSW Type</b>	
FLOAT_N	ECUC-ENUMERATION-LITERAL-DEF	
<b>BSW Description</b>		
Type of the data is float array.		
<b>Template Description</b>		
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.		
<b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.		
<b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency.  This attribute controls whether the data length of the data is fixed.		
<b>M2 Parameter</b>		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal/DcmDspArgumentScaling/DcmDspAlternativeArgumentData	
BSW Parameter		BSW Type
DcmDataElement		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
<p>Alternative Diagnosis Representation for the data defined by the means of a <code>ArgumentDataPrototype</code>.</p> <p>The <code>CompuMethod</code> of the data type of the referenced <code>ArgumentDataPrototype</code> will be applied to the data type of the <code>ArgumentDataPrototype</code> in the interface used by the <code>Dcm</code>.</p>		
Template Description		
An argument of an operation, much like a data element, but also carries direction information and is owned by a particular <code>ClientServerOperation</code> .		
M2 Parameter		
SWComponentTemplate::PortInterface::ArgumentDataPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_01056]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal/DcmDspArgumentScaling/DcmDspAlternativeDataType	
BSW Parameter		BSW Type
DcmApplicationDataType		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
<p>Alternative Diagnosis Representation for the data defined by the means of a <code>ApplicationDataType</code> of category <code>VALUE</code>, <code>BOOLEAN</code> or <code>ARRAY</code>.</p> <p>The <code>CompuMethod</code> that applies to the referenced <code>ApplicationDataType</code> in case of category <code>VALUE</code> or <code>BOOLEAN</code> will be applied to the data type of the <code>VariableDataPrototype</code> in the interface used by the <code>Dcm</code>.</p>		
Template Description		
A primitive data type defines a set of allowed values.		
M2 Parameter		
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_00998]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal/DcmDspRoutineSignalType	
BSW Parameter		BSW Type
FLOAT		ECUC-ENUMERATION-LITERAL-DEF





<b>BSW Description</b>	
Type of the data is float.	
<b>Template Description</b>	
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.	
<b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.	
<b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency.  This attribute controls whether the data length of the data is fixed.	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/ DcmDspStartRoutineOutSignal/DcmDspRoutineSignalType	
<b>BSW Parameter</b>	<b>BSW Type</b>	
FLOAT_N	ECUC-ENUMERATION-LITERAL-DEF	
<b>BSW Description</b>		
Type of the data is float array.		
<b>Template Description</b>		
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.		
<b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.		
<b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency.  This attribute controls whether the data length of the data is fixed.		
<b>M2 Parameter</b>		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine	
BSW Parameter	BSW Type	
DcmDspStopRoutine	ECUC-PARAM-CONF-CONTAINER-DEF	
BSW Description		
Provides the configuration of Stop subservice for RoutineControl service. Existence indicates that the StopRoutine in the RoutineControl is supported.		
Template Description		
<b>DiagnosticRoutine.stop:</b> This represents the ability to stop a running routine.		
<b>DiagnosticRoutineNeeds.diagRoutineType:</b> This denotes the type of diagnostic routine which is implemented by the referenced server port.		
M2 Parameter		
DiagnosticExtract::CommonDiagnostics::DiagnosticRoutine.stop, CommonStructure::ServiceNeeds::DiagnosticRoutineNeeds.diagRoutineType		
Mapping Rule		Mapping Type
1:1 mapping for DiagnosticRoutine.stop OR DiagnosticRoutineNeeds.diagRoutineType == asynchronous		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_01022]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal/DcmDspArgumentScaling/DcmDspAlternativeArgumentData	
BSW Parameter	BSW Type	
DcmDataElement	ECUC-FOREIGN-REFERENCE-DEF	
BSW Description		
Alternative Diagnosis Representation for the data defined by the means of a ArgumentDataPrototype.		
The CompuMethod of the data type of the referenced ArgumentDataPrototype will be applied to the data type of the ArgumentDataPrototype in the interface used by the Dcm.		
Template Description		
An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.		
M2 Parameter		
SWComponentTemplate::PortInterface::ArgumentDataPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_01056]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal/DcmDspArgumentScaling/DcmDspAlternativeData Type	







BSW Parameter		BSW Type
DcmApplicationDataType		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Alternative Diagnosis Representation for the data defined by the means of a ApplicationDataType of category VALUE, BOOLEAN or ARRAY. The CompuMethod that applies to the referenced ApplicationDataType in case of category VALUE or BOOLEAN will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.		
Template Description		
A primitive data type defines a set of allowed values.		
M2 Parameter		
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dcm_00998]

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal/DcmDspRoutineSignalType	
BSW Parameter		BSW Type
FLOAT		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Type of the data is float.		
Template Description		
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. <b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits. <b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeEncoding, AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeSize, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.fixedLength		
Mapping Rule		Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal/DcmDspRoutineSignalType	
BSW Parameter		BSW Type





FLOAT_N	ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>	
Type of the data is float array.	
<b>Template Description</b>	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeEncoding, AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeSize, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.fixedLength	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal/DcmDspArgumentScaling/DcmDspAlternativeArgumentData
<b>BSW Parameter</b>	<b>BSW Type</b>
DcmDataElement	ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>	
<p>Alternative Diagnosis Representation for the data defined by the means of a ArgumentDataPrototype.</p> <p>The CompuMethod of the data type of the referenced ArgumentDataPrototype will be applied to the data type of the ArgumentDataPrototype in the interface used by the Dcm.</p>	
<b>Template Description</b>	
An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.	
<b>M2 Parameter</b>	
SWComponentTemplate::PortInterface::ArgumentDataPrototype	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dcm_01056]

<b>BSW Module</b>	<b>BSW Context</b>
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/ DcmDspStopRoutineOutSignal/DcmDspArgumentScaling/DcmDspAlternativeData Type
<b>BSW Parameter</b>	<b>BSW Type</b>
DcmApplicationDataType	ECUC-FOREIGN-REFERENCE-DEF





BSW Description	
<p>Alternative Diagnosis Representation for the data defined by the means of a ApplicationDataType of category VALUE, BOOLEAN or ARRAY.</p> <p>The CompuMethod that applies to the referenced ApplicationDataType in case of category VALUE or BOOLEAN will be applied to the data type of the VariableDataPrototype in the interface used by the Dcm.</p>	
Template Description	
A primitive data type defines a set of allowed values.	
M2 Parameter	
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Dcm_00998]

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal/DcmDspRoutineSignalType
BSW Parameter	BSW Type
FLOAT	ECUC-ENUMERATION-LITERAL-DEF
BSW Description	
Type of the data is float.	
Template Description	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
M2 Parameter	
AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeEncoding, AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeSize, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.fixedLength	
Mapping Rule	Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
Mapping Status	ECUC Parameter ID
valid	

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal/DcmDspRoutineSignalType
BSW Parameter	BSW Type
FLOAT_N	ECUC-ENUMERATION-LITERAL-DEF
BSW Description	





Type of the data is float array.	
<b>Template Description</b>	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
<b>M2 Parameter</b>	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
<b>Mapping Rule</b>	<b>Mapping Type</b>
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity	
<b>BSW Parameter</b>	<b>BSW Type</b>	
DcmDspSecurityRow	ECUC-PARAM-CONF-CONTAINER-DEF	
<b>BSW Description</b>		
Definition of a single Row of configuration for security level configuration (per security level) The name of this container is used to define the name of the R-Port through which the DCM accesses the interface SecurityAccess_{SecurityLevel}. The R-Port is named SecurityAccess_{SecurityLevel} where {SecurityLevel} is the name of the container DcmDspSecurityRow. If there is no reference, no check of security level shall be done.		
<b>Template Description</b>		
This meta-class represents the needs of a software-component to verify the access to security level via diagnostic services.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::DiagnosticsCommunicationSecurityNeeds		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Dcm_00759]	

## G.6 Dem

<b>BSW Module</b>	<b>BSW Context</b>	
Dem	Dem/DemConfigSet	
<b>BSW Parameter</b>	<b>BSW Type</b>	
DemDebounceCounterBasedClass	ECUC-PARAM-CONF-CONTAINER-DEF	
<b>BSW Description</b>		





This container contains the configuration of Debounce Counter Based Class	
<b>Template Description</b>	
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.	
<b>M2 Parameter</b>	
CommonStructure::ServiceNeeds::DiagEventDebounceCounterBased	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dem_00881]

<b>BSW Module</b>	<b>BSW Context</b>	
Dem	Dem/DemConfigSet/DemDebounceCounterBasedClass	
<b>BSW Parameter</b>		<b>BSW Type</b>
DemDebounceCounterJumpDown		ECUC-BOOLEAN-PARAM-DEF
<b>BSW Description</b>		
Switch for the activation of Jump-Down. true: Jump-Down activated false: Jump-Down deactivated		
<b>Template Description</b>		
This value activates or deactivates the counter jump-down behavior.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::DiagEventDebounceCounterBased.counterJumpDown		
<b>Mapping Rule</b>		<b>Mapping Type</b>
Shall be taken from DiagnosticExtract::DiagnosticCommonProps.debounceAlgorithm Props.debounceAlgorithm.counterJumpDown. Applicable if DiagnosticExtract::DiagnosticCommon Props.debounceAlgorithmProps.debounceAlgorithm is modeled by means of a DiagEvent DebounceCounterBased.		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Dem_00685]

<b>BSW Module</b>	<b>BSW Context</b>	
Dem	Dem/DemConfigSet/DemDebounceCounterBasedClass	
<b>BSW Parameter</b>		<b>BSW Type</b>
DemDebounceCounterPassedThreshold		ECUC-INTEGER-PARAM-DEF
<b>BSW Description</b>		
Defines the value of the internal debounce counter, which indicates the passed status.		
<b>Template Description</b>		
This value defines the event-specific limit that indicates the "passed" counter status.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::DiagEventDebounceCounterBased.counterPassedThreshold		
<b>Mapping Rule</b>		<b>Mapping Type</b>





Shall be taken from DiagnosticExtract::DiagnosticCommonProps.debounceAlgorithmProps.debounceAlgorithm.counterPassedThreshold. Applicable if DiagnosticExtract::DiagnosticCommonProps.debounceAlgorithmProps.debounceAlgorithm is modeled by means of a DiagEventDebounceCounterBased.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dem_00636]

<b>BSW Module</b>	<b>BSW Context</b>	
Dem	Dem/DemConfigSet/DemEventParameter	
<b>BSW Parameter</b>		<b>BSW Type</b>
DemDebounceAlgorithmClass		ECUC-CHOICE-CONTAINER-DEF
<b>BSW Description</b>		
Debounce algorithm class: counter based, time based, or monitor internal.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::DiagEventDebounceAlgorithm		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Dem_00604]

<b>BSW Module</b>	<b>BSW Context</b>	
Dem	Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass	
<b>BSW Parameter</b>		<b>BSW Type</b>
DemDebounceCounterBased		ECUC-PARAM-CONF-CONTAINER-DEF
<b>BSW Description</b>		
This container contains the configuration (parameters) for counter based debouncing.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::DiagEventDebounceCounterBased		
<b>Mapping Rule</b>		<b>Mapping Type</b>
There are two ways to derive the existence of DemDebounceCounterBased: 1. DiagEventNeeds,diagEventDebounceAlgorithm exists and is modeled as a DiagEventDebounceCounterBased. 2. DiagnosticContributionSet.commonProperties.debounceAlgorithmProps.debounceAlgorithm exists and is modeled as a DiagEventDebounceCounterBased If both alternatives exist at the same time then the definition of DiagnosticContributionSet.commonProperties.debounceAlgorithmProps.debounceAlgorithm shall be handled with priority.		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Dem_00711]

BSW Module	BSW Context	
Dem	Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass	
BSW Parameter	BSW Type	
DemDebounceMonitorInternal	ECUC-PARAM-CONF-CONTAINER-DEF	
BSW Description		
This container contains the configuration (parameters) for monitor internal debouncing.		
Template Description		
"This meta-class represents the ability to indicate that no Dem pre-debounce algorithm shall be used for this diagnostic monitor. The SWC might implement an internal debouncing algorithm and report qualified (debounced) results to the Dem/DM.		
M2 Parameter		
CommonStructure::ServiceNeeds:: <a href="#">DiagEventDebounceMonitorInternal</a>		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dem_00712]

BSW Module	BSW Context	
Dem	Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass	
BSW Parameter	BSW Type	
DemDebounceTimeBase	ECUC-PARAM-CONF-CONTAINER-DEF	
BSW Description		
This container contains the configuration (parameters) for time based debouncing.		
Template Description		
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.		
M2 Parameter		
CommonStructure::ServiceNeeds:: <a href="#">DiagEventDebounceTimeBased</a>		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dem_00713]

BSW Module	BSW Context	
Dem	Dem/DemConfigSet/DemEventParameter	
BSW Parameter	BSW Type	
DemFFPrestorageInNvm	ECUC-BOOLEAN-PARAM-DEF	
BSW Description		
If the event uses a pre-stored freeze-frame this attribute indicates if the event requires the pre-stored data to be stored in non-volatile memory. TRUE = store the pre-stored data in non-volatile memory, FALSE = pre-stored data is not stored in non-volatile memory.		





Template Description	
<p><b>DiagnosticEventNeeds.prestoredFreezeFrameStoredInNvm:</b> If the Event uses a prestored freeze-frame (using the operations <code>PrestoreFreezeFrame</code> and <code>ClearPrestoredFreezeFrame</code> of the service interface <code>DiagnosticMonitor</code>) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm).</p> <p><b>DiagnosticEvent.prestoredFreezeFrameStoredInNvm:</b> If the Event uses a prestored freeze-frame (using the operations <code>PrestoreFreezeFrame</code> and <code>ClearPrestoredFreezeFrame</code> of the service interface <code>DiagnosticMonitor</code>) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm)</p>	
M2 Parameter	
CommonStructure::ServiceNeeds::DiagnosticEventNeeds.prestoredFreezeFrameStoredInNvm, Diagnostic Extract::Dem::DiagnosticEvent::DiagnosticEvent.prestoredFreezeFrameStoredInNvm	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Dem_00948]

BSW Module	BSW Context	
Dem	Dem/DemGeneral/DemDataElementClass/DemExternalCSDataElementClass/DemDataElementDataType	
BSW Parameter	BSW Type	
FLOAT	ECUC-ENUMERATION-LITERAL-DEF	
BSW Description		
Type of the data is float.		
Template Description		
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the <code>DiagnosticValueNeeds</code> is aggregated within a <code>BswModuleDependency</code>. This attribute controls whether the data length of the data is fixed.</p>		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeEncoding, AsamHdo::BaseTypes::BaseTypeDirectDefinition.baseTypeSize, CommonStructure::ServiceNeeds::DiagnosticValueNeeds.fixedLength		
Mapping Rule	Mapping Type	
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full	
Mapping Status	ECUC Parameter ID	
valid		

BSW Module	BSW Context	
Dem	Dem/DemGeneral/DemDataElementClass/DemExternalCSDataElementClass/DemDataElementDataType	







BSW Parameter		BSW Type
FLOAT_N		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Type of the data is float array.		
Template Description		
<b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.		
<b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.		
<b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency.  This attribute controls whether the data length of the data is fixed.		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
Mapping Rule		Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from DiagnosticValueNeeds.fixedLength=1 possible.		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
Dem	Dem/DemGeneral/DemDataElementClass/DemExternalCSDataElementClass	
BSW Parameter		BSW Type
DemDataElementProvideMonitorData		ECUC-BOOLEAN-PARAM-DEF
BSW Description		
If the parameter is set to True, the generated function call to retrieve the data element has the monitorData0 as additional first parameter.		
Template Description		
This attribute defines whether additional monitor data shall be added to the reporting of events.		
M2 Parameter		
CommonStructure::ServiceNeeds::DiagnosticEventNeeds. <a href="#">usesMonitorData</a>		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Dem_00951]

BSW Module	BSW Context	
Dem	Dem/DemGeneral/DemDataElementClass/DemExternalSRDataElementClass/DemDataElementDataType	
BSW Parameter		BSW Type
FLOAT		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Type of the data is float.		





Template Description	
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>	
M2 Parameter	
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>	
Mapping Rule	Mapping Type
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full
Mapping Status	ECUC Parameter ID
valid	

BSW Module	BSW Context	
Dem	Dem/DemGeneral/DemDataElementClass/DemExternalSRDataElementClass/DemDataElement DataType	
BSW Parameter	BSW Type	
FLOAT_N	ECUC-ENUMERATION-LITERAL-DEF	
BSW Description		
Type of the data is float array.		
Template Description		
<p><b>BaseTypeDirectDefinition.baseTypeEncoding:</b> This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence.</p> <p><b>BaseTypeDirectDefinition.baseTypeSize:</b> Describes the length of the data type specified in the container in bits.</p> <p><b>DiagnosticValueNeeds.fixedLength:</b> This attribute is applicable only if the DiagnosticValueNeeds is aggregated within a BswModuleDependency. This attribute controls whether the data length of the data is fixed.</p>		
M2 Parameter		
AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeEncoding</a> , AsamHdo::BaseTypes::BaseTypeDirectDefinition. <a href="#">baseTypeSize</a> , CommonStructure::ServiceNeeds::DiagnosticValueNeeds. <a href="#">fixedLength</a>		
Mapping Rule	Mapping Type	
baseTypeEncoding = NONE, WINDOWS-1252, UTF-8, BCD-P, BCD-UP baseTypeSize = 8 max NumberOfElements shall not exist arraySizeSemantics shall not exist Derivation from Diagnostic ValueNeeds.fixedLength=1 possible.	full	
Mapping Status	ECUC Parameter ID	
valid		

BSW Module	BSW Context	
Dem	Dem/DemGeneral/DemDataElementClass/DemExternalSRDataElementClass/DemDiagnosis Scaling/DemAlternativeDataInterface	





BSW Parameter		BSW Type	
DemDataElement		ECUC-FOREIGN-REFERENCE-DEF	
BSW Description			
<p>Alternative Diagnosis Representation for the data defined by the means of a VariableDataPrototype in a DataInterface.</p> <p>The CompuMethod of the data type of the referenced VariableDataPrototype will be applied to the data type of the VariableDataPrototype in the interface used by the Dem.</p>			
Template Description			
<p>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.</p> <p>In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.</p>			
M2 Parameter			
SWComponentTemplate::Datatype::DataPrototypes::VariableDataPrototype			
Mapping Rule		Mapping Type	
1:1 mapping		full	
Mapping Status		ECUC Parameter ID	
valid		[ECUC_Dem_00845]	

BSW Module		BSW Context	
Dem		Dem/DemGeneral/DemDataElementClass/DemExternalSRDataElementClass/DemDiagnosisScaling/DemAlternativeDataInterface	
BSW Parameter		BSW Type	
DemPortInterfaceMapping		ECUC-FOREIGN-REFERENCE-DEF	
BSW Description			
<p>Optional reference to PortInterfaceMapping which defines the mapping rules.</p> <p>The PortInterfaceMapping is used to get the DataPrototypeMapping that describes a conversion between the data prototype referenced by DemDataElement and the data prototype referenced from DcmDspExternalSRDataElementClass.</p>			
Template Description			
<p>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).</p>			
M2 Parameter			
SWComponentTemplate::PortInterface::PortInterfaceMapping			
Mapping Rule		Mapping Type	
1:1 mapping		full	
Mapping Status		ECUC Parameter ID	
valid		[ECUC_Dem_00846]	

BSW Module		BSW Context	
Dem		Dem/DemGeneral/DemDataElementClass/DemExternalSRDataElementClass/DemDiagnosisScaling/DemAlternativeDataType	
BSW Parameter		BSW Type	
DemApplicationDataType		ECUC-FOREIGN-REFERENCE-DEF	
BSW Description			





Alternative Diagnosis Representation for the data defined by the means of a ApplicationDataType of category VALUE, BOOLEAN or ARRAY.  The CompuMethod that applies to the referenced ApplicationDataType in case of category VALUE or BOOLEAN will be applied to the data type of the VariableDataPrototype in the interface used by the Dem.	
<b>Template Description</b>	
A primitive data type defines a set of allowed values.	
<b>M2 Parameter</b>	
SWComponentTemplate::Datatype::Datatypes::ApplicationPrimitiveDataType	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Dem_00848]

## G.7 BswM

<b>BSW Module</b>	<b>BSW Context</b>	
BswM	BswM/BswMConfig/BswMArbitration/BswMModeCondition/BswMConditionValue/BswMCompuScaleModeValue	
<b>BSW Parameter</b>		<b>BSW Type</b>
BswMCompuMethodRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
This is a foreign reference to the CompuMethod used for mode requests.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
AsamHdo::ComputationMethod::CompuMethod		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_BswM_01040]

<b>BSW Module</b>	<b>BSW Context</b>	
BswM	BswM/BswMConfig/BswMArbitration/BswMModeCondition/BswMConditionValue/BswMModeDeclaration	
<b>BSW Parameter</b>		<b>BSW Type</b>
BswMModeValueRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
This is a foreign reference to the Mode Declaration used for the mode requests corresponding to this condition.		
<b>Template Description</b>		
Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.		





M2 Parameter	
CommonStructure::ModeDeclaration::ModeDeclaration	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_BswM_00864]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMArbitration/BswMModeRequestPort/BswMModelInitValue/BswMCompuScaleModeValue	
BSW Parameter		BSW Type
BswMCompuMethodRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
This is a foreign reference to the CompuMethod used for mode requests.		
Template Description		
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.		
M2 Parameter		
AsamHdo::ComputationMethod::CompuMethod		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_BswM_01040]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMArbitration/BswMModeRequestPort/BswMModeRequestSource/BswMBswModeNotification	
BSW Parameter		BSW Type
BswMBswModeDeclarationGroupPrototypeRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
This is a foreign reference to the Mode Declaration Group Prototype.		
Template Description		
The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context.		
M2 Parameter		
CommonStructure::ModeDeclaration::ModeDeclarationGroupPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_BswM_00927]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMArbitration/BswMModeRequestPort/BswMModeRequestSource/BswMSwcModeNotification	
BSW Parameter		BSW Type
BswMSwcModeNotificationModeDeclarationGroupPrototypeRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
This is a foreign reference to the ModeDeclarationGroupPrototype.		
Template Description		
The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context.		
M2 Parameter		
CommonStructure::ModeDeclaration::ModeDeclarationGroupPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_BswM_00893]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMArbitration/BswMModeRequestPort/BswMModeRequestSource/BswMSwcModeRequest	
BSW Parameter		BSW Type
BswMSwcModeRequestVariableDataPrototypeRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
This is a reference to the VariableDataPrototype.		
Template Description		
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.		
M2 Parameter		
SWComponentTemplate::Datatype::DataPrototypes::VariableDataPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_BswM_01046]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMDataTypeMappingSets	
BSW Parameter		BSW Type
BswMDataTypeMappingSetRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Reference to DataTypeMappingSet.		





Template Description	
This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes. In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups.	
M2 Parameter	
SWComponentTemplate::Datatype::Datatypes::DataMappingSet	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_BswM_00937]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMModeControl/BswMAction/BswMAvailableActions/BswMRteModeRequest	
BSW Parameter		BSW Type
BswMRequestedModeRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
This is a foreign reference to the Mode Declaration used for the mode request		
Template Description		
Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.		
M2 Parameter		
CommonStructure::ModeDeclaration::ModeDeclaration		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_BswM_01024]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMModeControl/BswMAction/BswMAvailableActions/BswMRteSwitch	
BSW Parameter		BSW Type
BswMSwitchedMode		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
This parameter contains the integer value that corresponds to a certain mode in a Mode Declaration Group.		
Template Description		
Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.		
M2 Parameter		
CommonStructure::ModeDeclaration::ModeDeclaration		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_BswM_00896]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMModeControl/BswMAction/BswMAvailableActions/BswMSchMSwitch	
BSW Parameter		BSW Type
BswMSchMSwitchedMode		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
This parameter contains the integer value that corresponds to a certain mode in a Mode Declaration Group.		
Template Description		
Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.		
M2 Parameter		
CommonStructure::ModeDeclaration::ModeDeclaration		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_BswM_00901]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMModeControl/BswMRteModeRequestPort	
BSW Parameter		BSW Type
BswMRteModeRequestPortInterfaceRef		ECUC-INSTANCE-REFERENCE-DEF
BSW Description		
This is an instance reference to the variable data prototype used for the mode request.		
Template Description		
A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableData Prototype 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.		
M2 Parameter		
SWComponentTemplate::Datatype::DataPrototypes::VariableDataPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_BswM_01025]

BSW Module	BSW Context	
BswM	BswM/BswMConfig/BswMModeControl/BswMRteModeRequestPort	
BSW Parameter		BSW Type
BswMRteModeRequestVariableDataPrototypeSRRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
This is a foreign reference to a VariableDataPrototype used for the mode request.		
Template Description		







<p>A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableData Prototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided.</p> <p>In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.</p>	
<b>M2 Parameter</b>	
SWComponentTemplate::Datatype::DataPrototypes::VariableDataPrototype	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_BswM_01057]

<b>BSW Module</b>	<b>BSW Context</b>	
BswM	BswM/BswMConfig/BswMModeControl/BswMSwitchPort	
<b>BSW Parameter</b>		<b>BSW Type</b>
BswMModeSwitchInterfaceRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Reference to the ModeSwitchInterface from which the BswM will generate a PPortPrototype.		
<b>Template Description</b>		
A mode switch interface declares a ModeDeclarationGroupPrototype to be sent and received.		
<b>M2 Parameter</b>		
SWComponentTemplate::PortInterface::ModeSwitchInterface		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_BswM_00951]	

<b>BSW Module</b>	<b>BSW Context</b>	
BswM	BswM/BswMConfig/BswMModeControl/BswMSwitchPort	
<b>BSW Parameter</b>		<b>BSW Type</b>
BswMSchMModeDeclarationGroupRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Reference to the ModeDeclarationGroup from which the BswM will generate a ModeDeclarationGroupPrototype.		
<b>Template Description</b>		
A collection of Mode Declarations. Also, the initial mode is explicitly identified.		
<b>M2 Parameter</b>		
CommonStructure::ModeDeclaration::ModeDeclarationGroup		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_BswM_01031]	

## G.8 MemMap

BSW Module	BSW Context	
MemMap	MemMap/MemMapAllocation/MemMapGenericMapping	
BSW Parameter	BSW Type	
MemMapSwAddressMethodRef	ECUC-FOREIGN-REFERENCE-DEF	
BSW Description		
Reference to the SwAddrMethod which applies to the MemMapGenericMapping.		
Template Description		
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.		
M2 Parameter		
DataDictionary::AuxillaryObjects::SwAddrMethod		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_MemMap_00013]

BSW Module	BSW Context	
MemMap	MemMap/MemMapAllocation/MemMapSectionSpecificMapping	
BSW Parameter	BSW Type	
MemMapMemorySectionRef	ECUC-FOREIGN-REFERENCE-DEF	
BSW Description		
Reference to the MemorySection which applies to the MemMapSectionSpecificMapping.		
Template Description		
<p>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.</p> <p>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:</p> <pre>&lt;SwAddrMethod shortName&gt;[_&lt;further specialization nominator&gt;][_&lt;alignment&gt;]</pre> <p>where</p> <ul style="list-style-type: none"> <li>• [<b>&lt;SwAddrMethod shortName&gt;</b>] is the shortName of the referenced SwAddrMethod</li> <li>• [<b>&lt;further specialization nominator&gt;</b>] 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.</li> <li>• [<b>&lt;alignment&gt;</b>] is the alignment attributes value and is only applicable in the case that the memoryAllocation KeywordPolicy value of the referenced SwAddrMethod is set to addrMethodShortNameAndAlignment</li> </ul> <p>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.</p> <p>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.</p>		
M2 Parameter		





CommonStructure::ResourceConsumption::MemorySectionUsage::MemorySection	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_MemMap_-00016]

## G.9 RTE

<b>BSW Module</b>	<b>BSW Context</b>	
Rte	Rte/RteImplicitCommunication	
<b>BSW Parameter</b>		<b>BSW Type</b>
RteVariableReadAccessRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Reference to the VariableAccess in the dataReadAccess role.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
SWComponentTemplate::SwcInternalBehavior::DataElements::VariableAccess		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Rte_09035]

<b>BSW Module</b>	<b>BSW Context</b>	
Rte	Rte/RteImplicitCommunication	
<b>BSW Parameter</b>		<b>BSW Type</b>
RteVariableWriteAccessRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Reference to the VariableAccess in the dataWriteAccess role.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
SWComponentTemplate::SwcInternalBehavior::DataElements::VariableAccess		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Rte_09036]

BSW Module	BSW Context	
Rte	Rte/RteOsInteraction/RteModeToScheduleTableMapping/RteModeSchtblMapBsw	
BSW Parameter		BSW Type
RteModeSchtblMapBswProvidedModeGroupRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Reference to an instance of a ModeDeclarationGroupPrototype of a Bsw-Module.		
Template Description		
The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context.		
M2 Parameter		
CommonStructure::ModeDeclaration::ModeDeclarationGroupPrototype		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Rte_09053]

BSW Module	BSW Context	
Rte	Rte/RteOsInteraction/RteModeToScheduleTableMapping	
BSW Parameter		BSW Type
RteModeSchtblMapModeDeclarationRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Reference to the ModeDeclarations.		
Template Description		
Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.		
M2 Parameter		
CommonStructure::ModeDeclaration::ModeDeclaration		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Rte_09054]

BSW Module	BSW Context	
Rte	Rte/RteOsInteraction/RteModeToScheduleTableMapping/RteModeSchtblMapSwc	
BSW Parameter		BSW Type
RteModeSchtblMapSwcPortRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Reference to the PPortPrototype of a SwComponentPrototype.		
Template Description		
Component port providing a certain port interface.		
M2 Parameter		
SWComponentTemplate::Components::PPortPrototype		





Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Rte_09057]

BSW Module	BSW Context
Rte	Rte/RteSwComponentInstance/RteEventToIsrMapping
BSW Parameter	BSW Type
RteIsrEventRef	ECUC-FOREIGN-REFERENCE-DEF
BSW Description	
Reference to the description of the ExternalTriggerOccurredEvent or TimingEvent which is pointing to the RunnableEntity being mapped. This allows a fine grained mapping of RunnableEntites based on the activating RTEEvent.	
Template Description	
Abstract base class for all RTE-related events	
M2 Parameter	
SWComponentTemplate::SwcInternalBehavior::RTEEvents::RTEEvent	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Rte_09153]

BSW Module	BSW Context
Rte	Rte/RteSwComponentInstance/RteEventToTaskMapping
BSW Parameter	BSW Type
RteActivationOffset	ECUC-FLOAT-PARAM-DEF
BSW Description	
Activation offset in seconds.	
Template Description	
The value makes an assumption about the time offset of the first activation of the RunnableEntity triggered by the mapped TimingEvent relative to the periodic activation of the time base of this TimingEvent. Unit: second.	
M2 Parameter	
SWComponentTemplate::SwcInternalBehavior::RTEEvents::TimingEvent.offset	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_Rte_09018]

BSW Module	BSW Context
Rte	Rte/RteSwComponentInstance/RteEventToTaskMapping
BSW Parameter	BSW Type
RteEventRef	ECUC-FOREIGN-REFERENCE-DEF





<b>BSW Description</b>	
Reference to the description of the RTEEvent which is pointing to the RunnableEntity being mapped. This allows a fine grained mapping of RunnableEntites based on the activating RTEEvent.	
<b>Template Description</b>	
Abstract base class for all RTE-related events	
<b>M2 Parameter</b>	
SWComponentTemplate::SwcInternalBehavior::RTEEvents::RTEEvent	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Rte_09019]

<b>BSW Module</b>	<b>BSW Context</b>	
Rte	Rte/RteSwComponentInstance/RteExclusiveAreaImplementation	
<b>BSW Parameter</b>		<b>BSW Type</b>
RteExclusiveAreaRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Reference to the ExclusiveArea.		
<b>Template Description</b>		
Prevents an executable entity running in the area from being preempted.		
<b>M2 Parameter</b>		
CommonStructure::InternalBehavior::ExclusiveArea		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_Rte_09032]

<b>BSW Module</b>	<b>BSW Context</b>	
Rte	Rte/RteSwComponentInstance/RteInternalTriggerConfig	
<b>BSW Parameter</b>		<b>BSW Type</b>
RteSwcTriggerSourceRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Reference to an InternalTriggeringPoint of the related component instance. The referenced InternalTriggeringPoint has to belong to the same software component instance as the RteSwComponent Instance owning this parameter configures.		
<b>Template Description</b>		
If a RunnableEntity owns an InternalTriggeringPoint it is entitled to trigger the execution of RunnableEntities of the corresponding software-component.		
<b>M2 Parameter</b>		
SWComponentTemplate::SwcInternalBehavior::Trigger::InternalTriggeringPoint		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>





valid	[ECUC_Rte_09097]
-------	------------------

BSW Module	BSW Context	
Rte	Rte/RteSwComponentInstance/RteNvRamAllocation	
BSW Parameter	BSW Type	
RteSwNvBlockDescriptorRef	ECUC-FOREIGN-REFERENCE-DEF	
BSW Description		
Reference to the NvBlockDescriptor in case the RTE needs to call the NvM directly (e.g. for the supportDirtyFlag feature, storeCyclic feature, server invocation for NV data management or mode switch based invocation NvM services).		
Template Description		
Specifies the properties of exactly on NVRAM Block.		
M2 Parameter		
SWComponentTemplate::NvBlockComponent::NvBlockDescriptor		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Rte_09132]

BSW Module	BSW Context	
Rte	Rte/RteSwComponentInstance/RteNvRamAllocation	
BSW Parameter	BSW Type	
RteSwNvRamMappingRef	ECUC-FOREIGN-REFERENCE-DEF	
BSW Description		
Reference to the SwSeriveDependency which is used to specify the NvBlockNeeds.		
Template Description		
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.		
M2 Parameter		
SWComponentTemplate::SwcInternalBehavior::ServiceMapping::SwcServiceDependency		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Rte_09044]

BSW Module	BSW Context	
Rte	Rte/RteSwComponentInstance	
BSW Parameter	BSW Type	
RteSoftwareComponentInstanceRef	ECUC-FOREIGN-REFERENCE-DEF	
BSW Description		
Reference to a SwComponentPrototype.		
Template Description		





Role of a software component within a composition.	
<b>M2 Parameter</b>	
SWComponentTemplate::Composition::SwComponentPrototype	
<b>Mapping Rule</b>	<b>Mapping Type</b>
1:1 mapping	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_Rte_09004]

<b>BSW Module</b>	<b>BSW Context</b>	
Rte	Rte/RteSwComponentType/RteComponentTypeCalibration	
<b>BSW Parameter</b>		<b>BSW Type</b>
RteCalibrationSwAddrMethodRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Reference to the SwAddrMethod for which software calibration support shall be enabled.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
DataDictionary::AuxillaryObjects::SwAddrMethod		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Rte_09038]	

<b>BSW Module</b>	<b>BSW Context</b>	
Rte	Rte/RteSwComponentType	
<b>BSW Parameter</b>		<b>BSW Type</b>
RteComponentTypeRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Reference to either AtomicSwComponentType or ParameterSwComponentType.		
<b>Template Description</b>		
Base class for AUTOSAR software components.		
<b>M2 Parameter</b>		
SWComponentTemplate::Components::SwComponentType		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
1:1 mapping	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid	[ECUC_Rte_09003]	

<b>BSW Module</b>	<b>BSW Context</b>	
Rte	Rte/RteSwComponentType	







BSW Parameter		BSW Type	
RteImplementationRef		ECUC-FOREIGN-REFERENCE-DEF	
BSW Description			
The Implementation which shall be assigned to the SwComponentType.			
Template Description			
This meta-class represents a specialization of the general Implementation meta-class with respect to the usage in application software.			
M2 Parameter			
SWComponentTemplate::SwcImplementation::SwcImplementation			
Mapping Rule		Mapping Type	
1:1 mapping		full	
Mapping Status		ECUC Parameter ID	
valid		[ECUC_Rte_09028]	

## G.10 ECUC

BSW Module		BSW Context	
EcuC		EcuC/EcucConfigSet/EcucPduCollection/MetaDataType	
BSW Parameter		BSW Type	
MetaDataTypeItem		ECUC-PARAM-CONF-CONTAINER-DEF	
BSW Description			
The content of meta data in a Pdu consists of an ordered list of meta data items. This container represents a meta data item that is contained in meta data of a Pdu.			
Template Description			
This meta-class represents a single meta-data item.			
M2 Parameter			
SWComponentTemplate::PortInterface::MetaDataTypeItem			
Mapping Rule		Mapping Type	
1:1 mapping		full	
Mapping Status		ECUC Parameter ID	
valid		[ECUC_EcuC_00074]	

BSW Module		BSW Context	
EcuC		EcuC/EcucConfigSet/EcucPduCollection/MetaDataType/MetaDataTypeItem	
BSW Parameter		BSW Type	
MetaDataTypeItemLength		ECUC-INTEGER-PARAM-DEF	
BSW Description			
This parameter defines the length of a meta data item in bytes.			
Template Description			
This attribute determines the length of the MetaDataTypeItem at run-time.			
M2 Parameter			
SWComponentTemplate::PortInterface::MetaDataTypeItem.length			





Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	ECUC Parameter ID
valid	[ECUC_EcuC_00075]

BSW Module	BSW Context	
EcuC	EcuC/EcuConfigSet/EcuPduCollection/MetaDataType/MetaDataItem/MetaDataItemType	
BSW Parameter		BSW Type
ADDRESS_EXTENSION_8		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Address extension field (N_AE) of the mixed addressing modes with 11bit and 29bit CAN ID of ISO 15765-2. Size: 8 bits.		
Template Description		
This aggregation contributes the specification of the concrete meta-data item type.		
M2 Parameter		
SWComponentTemplate::PortInterface::MetaDataItem.metaDataItemType		
Mapping Rule		Mapping Type
SWComponentTemplate::PortInterface::MetaDataItem.metaDataItemType == ADDRESS_EXTENSION_16		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
EcuC	EcuC/EcuConfigSet/EcuPduCollection/MetaDataType/MetaDataItem/MetaDataItemType	
BSW Parameter		BSW Type
CAN_ID_32		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
CAN ID according to ISO 11898-2, either 29 bits or 11 bits. Encoding according to Can_IdType. Size: 32 bits.		
Template Description		
This aggregation contributes the specification of the concrete meta-data item type.		
M2 Parameter		
SWComponentTemplate::PortInterface::MetaDataItem.metaDataItemType		
Mapping Rule		Mapping Type
SWComponentTemplate::PortInterface::MetaDataItem.metaDataItemType == CAN_ID_32		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
EcuC	EcuC/EcuConfigSet/EcuPduCollection/MetaDataType/MetaDataItem/MetaDataItemType	
BSW Parameter		BSW Type
ETHERNET_MAC_64		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		





Ethernet MAC address. Size: 64 bits.	
<b>Template Description</b>	
This aggregation contributes the specification of the concrete meta-data item type.	
<b>M2 Parameter</b>	
SWComponentTemplate::PortInterface::MetaDatum.metaDataType	
<b>Mapping Rule</b>	<b>Mapping Type</b>
SWComponentTemplate::PortInterface::MetaDatum.metaDatumType == ETHERNET_MAC_64	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>	
EcuC	EcuC/EcuConfigSet/EcuPduCollection/MetaDataType/MetaDatum/MetaDatumType	
<b>BSW Parameter</b>		<b>BSW Type</b>
LIN_NAD_8		ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>		
LIN node address as used in the LIN transport protocol. Size: 8 bits.		
<b>Template Description</b>		
This aggregation contributes the specification of the concrete meta-data item type.		
<b>M2 Parameter</b>		
SWComponentTemplate::PortInterface::MetaDatum.metaDataType		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
SWComponentTemplate::PortInterface::MetaDatum.metaDatumType == LIN_NAD_8	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid		

<b>BSW Module</b>	<b>BSW Context</b>	
EcuC	EcuC/EcuConfigSet/EcuPduCollection/MetaDataType/MetaDatum/MetaDatumType	
<b>BSW Parameter</b>		<b>BSW Type</b>
PRIORITY_8		ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>		
Priority field of SAE J1939 IDs, or Ethernet QoS parameter. Size: 8 bits.		
<b>Template Description</b>		
This aggregation contributes the specification of the concrete meta-data item type.		
<b>M2 Parameter</b>		
SWComponentTemplate::PortInterface::MetaDatum.metaDataType		
<b>Mapping Rule</b>	<b>Mapping Type</b>	
SWComponentTemplate::PortInterface::MetaDatum.metaDatumType == PRIORITY_8	full	
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>	
valid		

BSW Module	BSW Context	
EcuC	EcuC/EcucConfigSet/EcucPduCollection/MetaDataType/MetaDataItem/MetaDataItemType	
BSW Parameter		BSW Type
SOCKET_CONNECTION_ID_16		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
SoAd socket connection ID. Size: 16 bits.		
Template Description		
This aggregation contributes the specification of the concrete meta-data item type.		
M2 Parameter		
SWComponentTemplate::PortInterface::MetaDataType.metaDataItemType		
Mapping Rule		Mapping Type
SWComponentTemplate::PortInterface::MetaDataType.metaDataItemType == SOCKET_CONNECTION_ID_16		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
EcuC	EcuC/EcucConfigSet/EcucPduCollection/MetaDataType/MetaDataItem/MetaDataItemType	
BSW Parameter		BSW Type
SOURCE_ADDRESS_16		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Source address of CanTp, FrTp, or DoIP transport protocol messages, or of SAE J1939 messages. Size: 16 bits.		
Template Description		
This aggregation contributes the specification of the concrete meta-data item type.		
M2 Parameter		
SWComponentTemplate::PortInterface::MetaDataType.metaDataItemType		
Mapping Rule		Mapping Type
SWComponentTemplate::PortInterface::MetaDataType.metaDataItemType == SOURCE_ADDRESS_16		full
Mapping Status		ECUC Parameter ID
valid		

BSW Module	BSW Context	
EcuC	EcuC/EcucConfigSet/EcucPduCollection/MetaDataType/MetaDataItem/MetaDataItemType	
BSW Parameter		BSW Type
TARGET_ADDRESS_16		ECUC-ENUMERATION-LITERAL-DEF
BSW Description		
Target address of CanTp, FrTp, or DoIP transport protocol messages, or destination address of SAE J1939 messages. Size: 16 bits.		
Template Description		
This aggregation contributes the specification of the concrete meta-data item type.		
M2 Parameter		





SWComponentTemplate::PortInterface::MetaDataTypeRef	
<b>Mapping Rule</b>	<b>Mapping Type</b>
SWComponentTemplate::PortInterface::MetaDataTypeRef == TARGET_ADDRESS_16	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	

<b>BSW Module</b>	<b>BSW Context</b>
EcuC	EcuC/EcucConfigSet/EcucPduCollection/Pdu
<b>BSW Parameter</b>	<b>BSW Type</b>
MetaDataTypeRef	ECUC-REFERENCE-DEF
<b>BSW Description</b>	
Reference to meta data that is transported in the Pdu through the AUTOSAR layers.	
<b>Template Description</b>	
<p><b>VariableDataPrototype:</b> 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.</p> <p>In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.</p> <p><b>SenderReceiverToSignalMapping:</b> Mapping of a sender receiver communication data element to a signal.</p> <p><b>SystemSignal:</b> 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.</p> <p><b>ISignal:</b> 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.</p> <p>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.</p> <p>ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping).</p> <p>In case of the SystemSignalGroup an ISignal shall be created for each SystemSignal contained in the SystemSignalGroup.</p> <p><b>ISignalIPdu:</b> 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.</p> <p>A maximum of one dynamic length signal per IPdu is allowed.</p>	
<b>M2 Parameter</b>	
SWComponentTemplate::Datatype::DataPrototypes::VariableDataPrototype, SystemTemplate::DataMapping::SenderReceiverToSignalMapping, SystemTemplate::Fibex::FibexCore::CoreCommunication::SystemSignal, SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal, SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignalIPdu	
<b>Mapping Rule</b>	<b>Mapping Type</b>
A MetaDataTypeRef shall be derived for a given Pdu if a MetaDataTypeSet exists that refers to a VariablePrototype that is also referenced from a SenderReceiverToSignalMapping that in turn references a SystemSignal that is referenced by a ISignal that is mapped to an ISignalIPdu that is derived to the mentioned Pdu in EcuC.	full
<b>Mapping Status</b>	<b>ECUC Parameter ID</b>
valid	[ECUC_EcuC_00077]

<b>BSW Module</b>	<b>BSW Context</b>	
EcuC	EcuC/EcucPartitionCollection/EcucPartition	
<b>BSW Parameter</b>		<b>BSW Type</b>
EcucPartitionSoftwareComponentInstanceRef		ECUC-INSTANCE-REFERENCE-DEF
<b>BSW Description</b>		
References the SW Component instances from the Ecu Extract that shall be executed in this partition.		
<b>Template Description</b>		
<b>M2 Parameter</b>		
SystemTemplate::SWmapping::SwcToEcuMapping.partition		
<b>Mapping Rule</b>		<b>Mapping Type</b>
The EcucPartitionSoftwareComponentInstanceRef is derived from an SwcToEcuMapping which references an EcuPartition and one or several SwComponentPrototypes. For each SwComponent Prototype that is referenced by the SwcToEcuMapping in the component role an EcucPartition SoftwareComponentInstanceRef shall be created that refers to the same SwComponentPrototype as the the SwcToEcuMapping.		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_EcuC_00036]

<b>BSW Module</b>	<b>BSW Context</b>	
EcuC	EcuC/EcucUnitGroupAssignment	
<b>BSW Parameter</b>		<b>BSW Type</b>
EcucUnitGroupRef		ECUC-FOREIGN-REFERENCE-DEF
<b>BSW Description</b>		
Optional reference to the UnitGroup to support the generation of ASAM MCD file. These UnitGroups are selecting a set of units for a specific country.		
<b>Template Description</b>		
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.		
<b>M2 Parameter</b>		
AsamHdo::Units::UnitGroup		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		[ECUC_EcuC_00062]

## G.11 OS

BSW Module	BSW Context	
Os	Os/Osloc/OslocCommunication/OslocDataProperties	
BSW Parameter		BSW Type
OslocDataTypeRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
<p>This is the type of the data to be transferred on the IOC communication channel. This attribute is necessary to generate the parameter type of the loc functions. Additionally this information should be used to compute the data size for necessary data copy operations within the loc module.</p> <p>If more than one attribute is defined, the IOC generator should generate an locXxxGroup function (Xxx= CHOICE [Send, Receive, Write, Read]).</p> <p>N:1 or N:M communication (Multiplicity of OslocSenderProperties &gt; 1) is only allowed for multiplicity of OslocDataTypeRef = 1</p>		
Template Description		
Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code.		
M2 Parameter		
CommonStructure::ImplementationDataTypes::ImplementationDataType		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Os_01005]

BSW Module	BSW Context	
Os	Os/Osloc/OslocCommunication/OslocDataProperties	
BSW Parameter		BSW Type
OsMemoryMappingCodeLocationRef		ECUC-FOREIGN-REFERENCE-DEF
BSW Description		
Reference to the memory mapping containing details about the section where the IOC buffer is placed.		
Template Description		
Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code.		
M2 Parameter		
CommonStructure::ImplementationDataTypes::ImplementationDataType		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		ECUC Parameter ID
valid		[ECUC_Os_00405]

## G.12 SecOC

BSW Module	BSW Context	
SecOC	SecOC/SecOCRxPduProcessing/SecOCClientServerVerificationStatusPropagationMode	
BSW Parameter		BSW Type





BOTH		ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>		
Both "TRUE" and "FALSE" AuthenticationStatus is propagated to SW-C		
<b>Template Description</b>		
Verification attempts that came out "false" or "true" shall be forwarded to the application software.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::VerificationStatusIndicationModeEnum. <a href="#">failureAndSuccess</a>		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
valid		

<b>BSW Module</b>	<b>BSW Context</b>	
SecOC	SecOC/SecOCRxPduProcessing/SecOCClientServerVerificationStatusPropagationMode	
<b>BSW Parameter</b>		<b>BSW Type</b>
FAILURE_ONLY		ECUC-ENUMERATION-LITERAL-DEF
<b>BSW Description</b>		
Only "FALSE" Authentication Status is propagated to SW-C		
<b>Template Description</b>		
Only verification attempts that came out "false" shall be forwarded to the application software.		
<b>M2 Parameter</b>		
CommonStructure::ServiceNeeds::VerificationStatusIndicationModeEnum. <a href="#">failureOnly</a>		
<b>Mapping Rule</b>		<b>Mapping Type</b>
1:1 mapping		full
<b>Mapping Status</b>		<b>ECUC Parameter ID</b>
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 the table 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 [11].

<i>Name of splitable element</i>	<i>Splitkey</i>
<a href="#">AdminData.sdq</a>	sdg, sdg.variationPoint.shortLabel
<a href="#">ARPackage.arPackage</a>	arPackage.shortName, arPackage.variationPoint.shortLabel
<a href="#">ARPackage.element</a>	element.shortName, element.definition, element.variationPoint.shortLabel
<a href="#">ARPackage.referenceBase</a>	referenceBase.shortLabel
<a href="#">AtomicSwComponentType.internalBehavior</a>	internalBehavior.shortName, internalBehavior.variationPoint.shortLabel
<a href="#">AtomicSwComponentType.symbolProps</a>	symbolProps.shortName
<a href="#">CompositionSwComponentType.component</a>	component.shortName, component.variationPoint.shortLabel
<a href="#">CompositionSwComponentType.connector</a>	connector.shortName, connector.variationPoint.shortLabel
<a href="#">CompositionSwComponentType.constantValueMapping</a>	constantValueMapping
<a href="#">CompositionSwComponentType.dataTypeMapping</a>	dataTypeMapping
<a href="#">CompositionSwComponentType.instantiationRTEEventProps</a>	instantiationRTEEventProps.shortLabel, instantiationRTEEventProps.variationPoint.shortLabel
<a href="#">ConsistencyNeeds.dpgDoesNotRequireCoherency</a>	dpgDoesNotRequireCoherency.shortName, dpgDoesNotRequireCoherency.variationPoint.shortLabel
<a href="#">ConsistencyNeeds.dpgRequiresCoherency</a>	dpgRequiresCoherency.shortName, dpgRequiresCoherency.variationPoint.shortLabel
<a href="#">ConsistencyNeeds.regDoesNotRequireStability</a>	regDoesNotRequireStability.shortName, regDoesNotRequireStability.variationPoint.shortLabel
<a href="#">ConsistencyNeeds.regRequiresStability</a>	regRequiresStability.shortName, regRequiresStability.variationPoint.shortLabel
<a href="#">Describable.adminData</a>	adminData
<a href="#">EndToEndProtection.endToEndProfile</a>	endToEndProfile
<a href="#">EndToEndProtection.endToEndProtectionISignalIPdu</a>	endToEndProtectionISignalIPdu, endToEndProtectionISignalIPdu.variationPoint.shortLabel
<a href="#">EndToEndProtection.endToEndProtectionVariablePrototype</a>	endToEndProtectionVariablePrototype.shortLabel, endToEndProtectionVariablePrototype.variationPoint.shortLabel
<a href="#">EndToEndProtectionSet.endToEndProtection</a>	endToEndProtection.shortName, endToEndProtection.variationPoint.shortLabel
<a href="#">Identifiable.adminData</a>	adminData
<a href="#">Implementation.mcSupport</a>	mcSupport
<a href="#">Implementation.resourceConsumption</a>	resourceConsumption.shortName
<a href="#">ImplementationDataType.symbolProps</a>	symbolProps.shortName
<a href="#">InternalBehavior.constantMemory</a>	constantMemory.shortName, constantMemory.variationPoint.shortLabel





<b>Name of splittable element</b>	<b>Splitkey</b>
InternalBehavior.constantValueMapping	constantValueMapping
InternalBehavior.dataTypeMapping	dataTypeMapping
InternalBehavior.exclusiveArea	exclusiveArea.shortName, exclusiveArea.variationPoint.shortLabel
InternalBehavior.exclusiveAreaNestingOrder	exclusiveAreaNestingOrder.shortName, exclusiveAreaNestingOrder.variationPoint.shortLabel
InternalBehavior.staticMemory	staticMemory.shortName, staticMemory.variationPoint.shortLabel
NvBlockDescriptor.constantValueMapping	constantValueMapping
NvBlockDescriptor.dataTypeMapping	dataTypeMapping
NvBlockDescriptor.modeSwitchEventTriggeredActivity	modeSwitchEventTriggeredActivity, modeSwitchEventTriggeredActivity.variationPoint.shortLabel
NvBlockSwComponentType.bulkNvDataDescriptor	bulkNvDataDescriptor.shortName, bulkNvDataDescriptor.variationPoint.shortLabel
NvBlockSwComponentType.nvBlockDescriptor	nvBlockDescriptor.shortName, nvBlockDescriptor.variationPoint.shortLabel
ParameterSwComponentType.constantMapping	constantMapping
ParameterSwComponentType.dataTypeMapping	dataTypeMapping
RapidPrototypingScenario.rptContainer	rptContainer.shortName, rptContainer.variationPoint.shortLabel
RapidPrototypingScenario.rptProfile	rptProfile.shortName
RapidPrototypingScenario.rptSystem	rptSystem
RptContainer.byPassPoint	byPassPoint.contextElement, byPassPoint.target, byPassPoint.variationPoint.shortLabel
RptContainer.explicitRptProfileSelection	explicitRptProfileSelection
RptContainer.rptContainer	rptContainer.shortName, rptContainer.variationPoint.shortLabel
RptContainer.rptHook	rptHook, rptHook.variationPoint.shortLabel
RTEEvent.disabledMode	disabledMode.contextPort, disabledMode.contextModeDeclarationGroupPrototype, disabledMode.targetModeDeclaration
RunnableEntity.asynchronousServerCallResultPoint	asynchronousServerCallResultPoint.shortName, asynchronousServerCallResultPoint.variationPoint.shortLabel
RunnableEntity.dataReadAccess	dataReadAccess.shortName, dataReadAccess.variationPoint.shortLabel
RunnableEntity.dataReceivePointByArgument	dataReceivePointByArgument.shortName, dataReceivePointByArgument.variationPoint.shortLabel
RunnableEntity.dataReceivePointByValue	dataReceivePointByValue.shortName, dataReceivePointByValue.variationPoint.shortLabel
RunnableEntity.dataSendPoint	dataSendPoint.shortName, dataSendPoint.variationPoint.shortLabel
RunnableEntity.dataWriteAccess	dataWriteAccess.shortName, dataWriteAccess.variationPoint.shortLabel
RunnableEntity.externalTriggeringPoint	externalTriggeringPoint.ident.shortName, externalTriggeringPoint.variationPoint.shortLabel
RunnableEntity.internalTriggeringPoint	internalTriggeringPoint.shortName, internalTriggeringPoint.variationPoint.shortLabel
RunnableEntity.modeAccessPoint	modeAccessPoint.ident.shortName, modeAccessPoint.variationPoint.shortLabel
RunnableEntity.modeSwitchPoint	modeSwitchPoint.shortName, modeSwitchPoint.variationPoint.shortLabel





<b>Name of splitable element</b>	<b>Splitkey</b>
RunnableEntity.parameterAccess	parameterAccess.shortName, parameterAccess.variationPoint.shortLabel
RunnableEntity.readLocalVariable	readLocalVariable.shortName, readLocalVariable.variationPoint.shortLabel
RunnableEntity.serverCallPoint	serverCallPoint.shortName, serverCallPoint.variationPoint.shortLabel
RunnableEntity.writtenLocalVariable	writtenLocalVariable.shortName, writtenLocalVariable.variationPoint.shortLabel
SwcInternalBehavior.arTypedPerInstanceMemory	arTypedPerInstanceMemory.shortName, arTypedPerInstanceMemory.variationPoint.shortLabel
SwcInternalBehavior.event	event.shortName, event.variationPoint.shortLabel
SwcInternalBehavior.exclusiveAreaPolicy	exclusiveAreaPolicy, exclusiveAreaPolicy.variationPoint.shortLabel
SwcInternalBehavior.explicitInterRunnableVariable	explicitInterRunnableVariable.shortName, explicitInterRunnableVariable.variationPoint.shortLabel
SwcInternalBehavior.implicitInterRunnableVariable	implicitInterRunnableVariable.shortName, implicitInterRunnableVariable.variationPoint.shortLabel
SwcInternalBehavior.includedDataTypeSet	includedDataTypeSet
SwcInternalBehavior.includedModeDeclarationGroupSet	includedModeDeclarationGroupSet
SwcInternalBehavior.instantiationDataDefProps	instantiationDataDefProps, instantiationDataDefProps.variationPoint.shortLabel
SwcInternalBehavior.perInstanceMemory	perInstanceMemory.shortName, perInstanceMemory.variationPoint.shortLabel
SwcInternalBehavior.perInstanceParameter	perInstanceParameter.shortName, perInstanceParameter.variationPoint.shortLabel
SwcInternalBehavior.portAPIOption	portAPIOption, portAPIOption.variationPoint.shortLabel
SwcInternalBehavior.runnable	runnable.shortName, runnable.variationPoint.shortLabel
SwcInternalBehavior.serviceDependency	serviceDependency.shortName, serviceDependency.variationPoint.shortLabel
SwcInternalBehavior.sharedParameter	sharedParameter.shortName, sharedParameter.variationPoint.shortLabel
SwcInternalBehavior.variationPointProxy	variationPointProxy.shortName
SwComponentType.consistencyNeeds	consistencyNeeds.shortName, consistencyNeeds.variationPoint.shortLabel
SwComponentType.port	port.shortName, port.variationPoint.shortLabel
SwComponentType.swComponentDocumentation	swComponentDocumentation, swComponentDocumentation.variationPoint.shortLabel
SwcServiceDependency.assignedPort	assignedPort, assignedPort.variationPoint.shortLabel

**Table H.1: Usage of splitable 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 the table consists of the identification of the model element itself and the applicable value of the tagged value `vh.latestBindingTime`.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [11].

<i>Variation Point</i>	<i>Latest Binding Time</i>
<a href="#">ApplicationArrayElement.maxNumberOfElements</a>	preCompileTime
<a href="#">ApplicationRecordDataType.element</a>	preCompileTime
<a href="#">ARPackage.arPackage</a>	blueprintDerivationTime
<a href="#">ARPackage.element</a>	systemDesignTime
<a href="#">ArrayValueSpecification.element</a>	preCompileTime
<a href="#">AtomicSwComponentType.internalBehavior</a>	preCompileTime
<a href="#">BulkNvDataDescriptor.nvBlockDataMapping</a>	preCompileTime
<a href="#">CalibrationParameterValueSet.calibrationParameterValue</a>	preCompileTime
<a href="#">ClientServerInterface.operation</a>	blueprintDerivationTime
<a href="#">ClientServerOperation.argument</a>	blueprintDerivationTime
<a href="#">CompositionSwComponentType.component</a>	postBuild
<a href="#">CompositionSwComponentType.connector</a>	postBuild
<a href="#">CompositionSwComponentType.instantiationRTEEventProps</a>	codeGenerationTime
<a href="#">CompuConstFormulaContent.vf</a>	codeGenerationTime
<a href="#">CompuNominatorDenominator.v</a>	preCompileTime
<a href="#">CompuScale.lowerLimit</a>	preCompileTime
<a href="#">CompuScale.upperLimit</a>	preCompileTime
<a href="#">CompuScales.compuScale</a>	blueprintDerivationTime
<a href="#">ConsistencyNeeds.dpgDoesNotRequireCoherency</a>	preCompileTime
<a href="#">ConsistencyNeeds.dpgRequiresCoherency</a>	preCompileTime
<a href="#">ConsistencyNeeds.regDoesNotRequireStability</a>	preCompileTime
<a href="#">ConsistencyNeeds.regRequiresStability</a>	preCompileTime
<a href="#">DataPrototypeGroup.dataPrototypeGroup</a>	preCompileTime
<a href="#">DataPrototypeGroup.implicitDataAccess</a>	preCompileTime
<a href="#">DiagEventDebounceCounterBased.counterDecrementStepSize</a>	preCompileTime
<a href="#">DiagEventDebounceCounterBased.counterFailedThreshold</a>	preCompileTime
<a href="#">DiagEventDebounceCounterBased.counterIncrementStepSize</a>	preCompileTime
<a href="#">DiagEventDebounceCounterBased.counterJumpDown</a>	preCompileTime
<a href="#">DiagEventDebounceCounterBased.counterJumpDownValue</a>	preCompileTime
<a href="#">DiagEventDebounceCounterBased.counterJumpUp</a>	preCompileTime
<a href="#">DiagEventDebounceCounterBased.counterJumpUpValue</a>	preCompileTime
<a href="#">DiagEventDebounceCounterBased.counterPassedThreshold</a>	preCompileTime
<a href="#">DiagEventDebounceTimeBased.timeBasedFdcThresholdStorageValue</a>	preCompileTime
<a href="#">DiagEventDebounceTimeBased.timeFailedThreshold</a>	preCompileTime
<a href="#">DiagEventDebounceTimeBased.timePassedThreshold</a>	preCompileTime





<b>Variation Point</b>	<b>Latest Binding Time</b>
EndToEndProtection.endToEndProtectionISignalPdu	preCompileTime
EndToEndProtection.endToEndProtectionVariablePrototype	preCompileTime
EndToEndProtectionSet.endToEndProtection	preCompileTime
ErrorTracerNeeds.tracedFailure	preCompileTime
ExecutableEntity.canEnter	preCompileTime
ExecutableEntity.runsInside	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.modeSwitchEventTriggeredActivity	preCompileTime
NvBlockDescriptor.nvBlockDataMapping	preCompileTime
NvBlockSwComponentType.bulkNvDataDescriptor	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.vtf	preCompileTime
RuleBasedValueSpecification.arguments	preCompileTime
RunnableEntity.asynchronousServerCallResultPoint	preCompileTime





<b>Variation Point</b>	<b>Latest Binding Time</b>
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
SupervisedEntityNeeds.checkpoints	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.exclusiveAreaPolicy	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





<b>Variation Point</b>	<b>Latest Binding Time</b>
SwcServiceDependency.assignedData	preCompileTime
SwcServiceDependency.assignedPort	preCompileTime
SwDataDefProps	codeGenerationTime
SwDataDefProps.swValueBlockSize	preCompileTime
SwDataDefProps.swValueBlockSizeMult	preCompileTime
SwGenericAxisParam.vf	preCompileTime
SwTextProps.swMaxTextSize	preCompileTime
SwValues.vf	preCompileTime
SwValues.vtf	preCompileTime
TextTableMapping.bitfieldTextTableMaskFirst	preCompileTime
TextTableMapping.bitfieldTextTableMaskSecond	preCompileTime
TextTableValuePair.firstValue	preCompileTime
TextTableValuePair.secondValue	preCompileTime
ValueList.vf	preCompileTime

**Table I.1: Usage of variation points**