Document Title	Standardization Template
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	535

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R25-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2025-11-27	R25-11	AUTOSAR Release Management	• Taken over content from TR_PredefinedNames • Corrections to traceable rules/conventions • Removal of PDEP • Add Trace groups description
2024-11-27	R24-11	AUTOSAR Release Management	• Obsolete categories of traceable removed in TPS_STDT_00098
2023-11-23	R23-11	AUTOSAR Release Management	• Add chapter on AUTOSAR Imposition Times • Remove SAFEX traceable item category • Various editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Advisory item • Sentence pattern • Extension of namePattern for platform
2021-11-25	R21-11	AUTOSAR Release Management	• update life cycle states • improve traceability to RS document
2020-11-30	R20-11	AUTOSAR Release Management	• introduce advisory markup • editorial changes • Migration of document to standard FO

▽

△

2019-11-28	R19-11	AUTOSAR Release Management	• harmonize the use of BlueprintCondition, FormalBlueprintGenerator • editorial changes • changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• uptraces wrt. life cycles • include ARMQL relevant parts • harmonize Blueprint parts
2017-12-08	4.3.1	AUTOSAR Release Management	• editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• extend Blueprintables • update specification levels • convert constraints in specification items • introduction of platform based document structure • introduction of Profiles for Data Exchange Points
2015-07-31	4.2.2	AUTOSAR Release Management	• introduction of LifeCycleState for constraint and specification items • editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• introduction of Blueprint Policy • include safety extension relevant items • extension of acceptance test items
2014-03-31	4.1.3	AUTOSAR Release Management	• editorial changes including tagged specification items • update content of specification levels
2013-10-31	4.1.2	AUTOSAR Release Management	• editorial changes including tagged specification items • extension of blueprinting to further AUTOSAR classes

▽

△

2013-03-15	4.1.1	AUTOSAR Administration	• editorial changes including tagged specification items • extension of blueprinting to further AUTOSAR classes (e.g. build action manifest) • introduction of life cycle support • improvement of document traceability • refinement of traceability support
2011-12-22	4.0.3	AUTOSAR Administration	• Initial Release

Disclaimer

Table of Contents

References

[1] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[2] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

[3] XML Specification of Application Interfaces
AUTOSAR_CP_MOD_AISpecification

[4] Technical Report on AUTOSAR Features
AUTOSAR_FO_TR_Features

[5] AUTOSAR Feature Model
AUTOSAR_FO_MOD_Features

[6] Collection of blueprints for AUTOSAR M1 models
AUTOSAR_FO_MOD_GeneralBlueprints

[7] Standardized M1 Models used for the Definition of AUTOSAR
AUTOSAR_FO_MOD_GeneralDefinitions

[8] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General

[9] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[10] Requirements on BSW Modules for SAE J1939
AUTOSAR_CP_RS_SAEJ1939

[11] Specification of a Transport Layer for SAE J1939
AUTOSAR_CP_SWS_SAEJ1939TransportLayer

[12] Specification of a Request Manager for SAE J1939
AUTOSAR_CP_SWS_SAEJ1939RequestManager

[13] XML Path language (XPath)
http://www.w3.org/TR/xpath/

[14] ANTLR parser generator V3
http://www.antlr.org

[15] Specification of ECU Configuration
AUTOSAR_CP_TPS_ECUConfiguration

[16] Modeling and Naming Aspects for Documentation, Measurement, and Calibration
AUTOSAR_CP_TR_AIMeasurementCalibrationDiagnostics

[17] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[18] Specification of Timing Extensions for Classic Platform
AUTOSAR_CP_TPS_TimingExtensions

[19] Explanation of Application Interfaces of the Powertrain Engine Domain
AUTOSAR_CP_EXP_AIPowertrain

[20] Specification of Platform Types for Classic Platform
AUTOSAR_CP_SWS_PlatformTypes

[21] SW-C and System Modeling Guide
AUTOSAR_CP_TR_SWCModelingGuide

1 Introduction

The document describes aspects related to published AUTOSAR documents. The types of documents and their relation; how to read them; the usage of the AUTOSAR meta-model for AUTOSAR documentation and in particular tracing in AUTOSAR documents.

1.1 AUTOSAR document conventions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The headers in the table have the following meaning:

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

Type: The type of an attribute of the class.

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

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

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

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

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

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

2 AUTOSAR documentation model

2.1 AUTOSAR document meta-data

The table in [TPS_STDT_00151] lists the meta-data for any given AUTOSAR document, the "Example" column shows the meta-data for this document.

[TPS_STDT_00151] AUTOSAR document meta-data ⌈

Meta-data attribute	Mnemonic	Usage	Example
Document category (full)	doc-cat-full	Document category - full name	Template Specification
Document category (abbr)	doc-cat-abbr	Document category - abbreviated name	TPS
Document PDF name	doc-file-name	Document published .pdf file name	AUTOSAR_FO_TPS_StandardizationTemplate
Document identifier	doc-id	Document unique identifier	535
Document standard (full)	doc-std-full	AUTOSAR standard - full name. If unspecified, the **"Part of AUTOSAR Standard"** meta-data applies implicitly	Foundation
Document standard (abbr)	doc-std-abbr	Abbreviated form of doc-std-full. As per StandardNameEnum.	FO
Document status	doc-status	Document publishing status	published
Document release	doc-rel-ver	AUTOSAR release in which the document is published	R25-11
Document name (full)	doc-name-full	Document name (full) - from [TPS_STDT_00137]	StandardizationTemplate
Document name (abbr)	doc-name-abbr	Document name (abbreviated) - from [TPS_STDT_00137]	STDT
Document title	doc-title	Document title	Standardization Template

⌋

2.2 Document categories

The table in [TPS_STDT_00150] lists the different categories of documents published by AUTOSAR[1] and their purpose.

[TPS_STDT_00150] AUTOSAR document categories ⌈

Document category (abbreviation)	Document category (full)	Purpose
ASWS	Abstract SWS Software Specification	General Specification of AUTOSAR Basic Software Modules
EXP	Explanation	Explanatory material discussing contents already shown in other documents

▽

[1] Strictly speaking, MOD, MMOD, SRC are not documents, rather 'other' formats of published artifacts

△

Document category (abbreviation)	Document category (full)	Purpose
MMOD	MetaModel	Modeled contents (a model or generated from a model) on meta level 2 (Meta-Model)
MOD	Model	Modeled contents (a model or generated from a model) on meta level 1 (Model)
PRS	Protocol Specification	Specification of Protocols standardized by AUTOSAR
RS	Requirement Specification	Specification of requirements other than for software specifications
SRC	Source	Source code artifacts
SWS	Software Specification	Specification of AUTOSAR Software
TPS	Template Specification	Specification of AUTOSAR Templates, containing Meta model information, constraints etc.
TR	Technical Report	A general technical report describing arbitrary AUTOSAR related topics

⌋

2.3 Document relational views

Documents are published in one of the given `standard` contexts. Figure 2.1 shows the placement of an AUTOSAR document category in a respective `standard`.

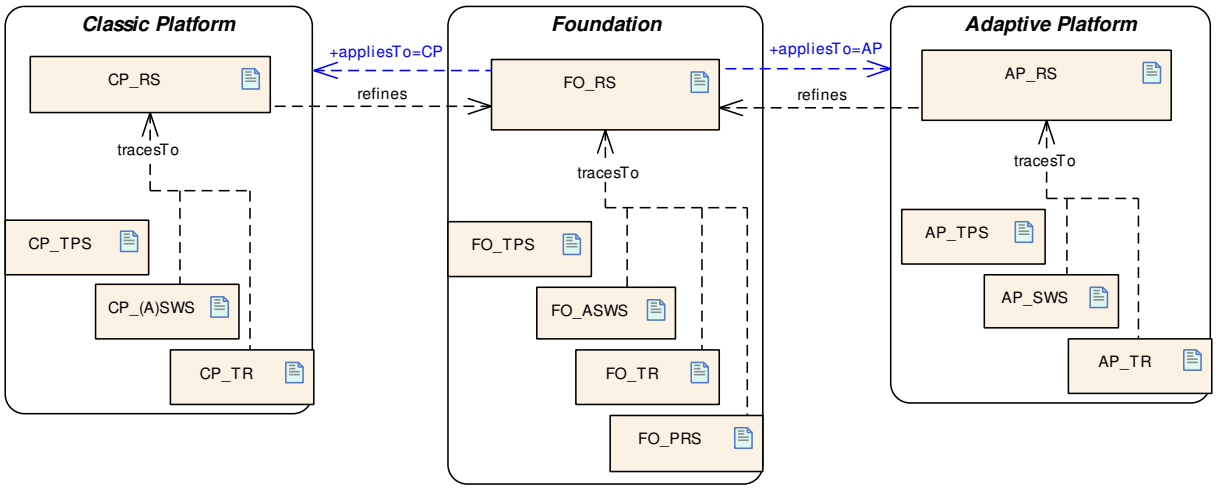


Figure 2.1: AUTOSAR document category vs. AUTOSAR standard

In 2.2 it is shown the high-level relation between the AUTOSAR document types.

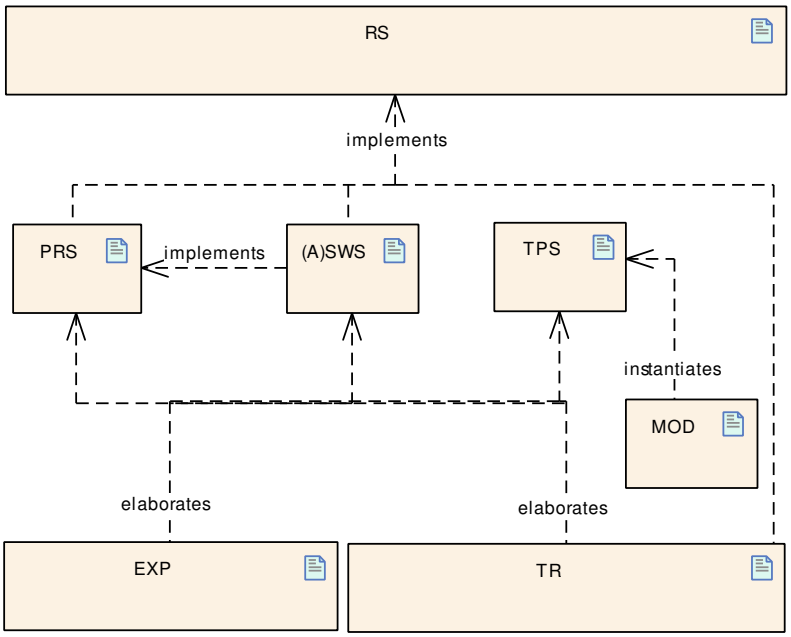


Figure 2.2: AUTOSAR inter-document relations

In general the relations are:

- «implements»: a `PRS`, `(A)SWS` implement requirements from a `RS`; an `SWS` implements specification items from a `PRS`
- «elaborates»: a `TR`, `EXP` further explain or further extend technical content from another specification
- «instantiates»: an (M1) `MOD` instantiates a (M2) `TPS` (model) (see [2] Chapter 2.2 *"The AUTOSAR meta-model hierarchy"*)

Tracing relations between documents are elaborated in chapter 3.2.

2.4 Document list

The table in [TPS_STDT_00137] lists the suite of documents published by AUTOSAR.

[TPS_STDT_00137] AUTOSAR Document Names and Abbreviations for Trace Prefixes ⌈

Document name (full)	***Document name (abbreviation)***	***Document title***
ADCDriver	Adc	Requirements on ADC Driver; Specification of ADC Driver
AIADASAndVMC	n/a	Explanation of Application Interface of AD/ADAS vehicle motion control
AIBodyAndComfort	AIBC	Explanation of Application Interfaces of the Body and Comfort Domain

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
AIChassis	AICS	Explanation of Application Interfaces of the Chassis Domain
AIDesignPatternsCatalogue	n/a	Application Design Patterns Catalogue
AIHMIMultimediaAndTelematics	n/a	Explanation of Application Interfaces of the HMI, Multimedia and Telematics Domain
AIMeasurementCalibrationDiagnostics	MCR; MCA; MCG; MCM	Modeling and Naming Aspects for Documentation, Measurement, and Calibration
AIOccupantAndPedestrianSafety	AIOPS	Explanation of Application Interfaces of Occupant and Pedestrian Safety Systems Domain
AIPowertrain	AIPT	Explanation of Application Interfaces of the Powertrain Engine Domain
AISpecification	n/a	XML Specification of Application Interfaces
AISpecificationExamples	n/a	Application Interface Examples
AIUserGuide	AIUG	Application Interfaces User Guide
ARAComAPI	n/a	Explanation of ara::com API
ARAHeaderFiles	n/a	Source of Adaptive Platform ARAHeaderFiles
ARARustApplications	n/a	Explanation of ARA Applications in Rust
ARTI	Arti	Explanatory Document for Usage of AUTOSAR Run-Time Interface; Specification of AUTOSAR Run-Time Interface
ARXMLSerializationRules	ASR	ARXML Serialization Rules
AbstractPlatformSpecification	APSD	Specification of Abstract Platform
ApplicationLevelErrorHandling	ALEH	Explanation of Error Handling on Application Level
AutomatedDrivingInterfaces	ADI	Requirements on Automated Driving Interfaces
AutomotiveAPI	n/a	Explanation of Automotive API
AutomotiveAPIGateway	AAG	Requirements on Automotive API Gateway; Specification of Automotive API Gateway
AutosarModelConstraints	n/a	Collection of constraints on AUTOSAR M1 models
BFXLibrary	Bfx	Specification of Bit Handling Library
BSWDistributionGuide	n/a	Guide to BSW Distribution
BSWGeneral	BSW	General Requirements on Basic Software Modules; General Specification of Basic Software Modules
BSWModeManager	BswM	Specification of Basic Software Mode Manager
BSWModuleDescriptionTemplate	BSWMDT	Basic Software Module Description Template
BSWMulticoreLibrary	BMC	Specification of Basic Software Multicore Library
BSWUMLModel	n/a	Basic Software UML Model
BSWUMLModelModelingGuide	BSWMG	Modeling Guidelines of Basic Software EA UML Model
BulkNvDataManager	BndM	Specification of Bulk NvData Manager
BusMirroring	Mirror	Requirements on Bus Mirroring; Specification of Bus Mirroring
CAN	Can	Requirements on CAN
CANDriver	Can	Specification of CAN Driver
CANInterface	CANIF	Specification of CAN Interface
CANNetworkManagement	CanNm	Specification of CAN Network Management
CANStateManager	CanSM	Specification of CAN State Manager
CANTransceiverDriver	CanTrcv	Specification of CAN Transceiver Driver

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
CANTransportLayer	CanTp	Specification of CAN Transport Layer
CANXLDriver	CanXL	Specification for CAN XL Driver
CANXLTransceiverDriver	CanXLTrcv	Specification of CAN XL Transceiver Driver
CDDDesignAndIntegrationGuideline	n/a	Complex Driver design and integration guideline
COM	Com	Requirements on Communication; Specification of Communication
COMBasedTransformer	ComXf	Specification of COM Based Transformer
COMManager	ComM	Specification of Communication Manager
CRCLibrary	Crc	Specification of CRC Library
CellularV2XDriver	CV2x	Specification of Cellular Vehicle-2-X Driver
ChangeDocumentation	n/a	Adaptive Platform Change Documentation; Classic Platform Change Documentation; Foundation Change Documentation
ChargingManager	ChrgM	Requirements on Charging Manager
ChineseV2XCommunication	CnV2X	Requirements on Chinese Vehicle-2-X Communication
ChineseV2XManagement	CnV2xM	Specification of Chinese Vehicle-2-X Management
ChineseV2XMessage	CnV2xMsg	Specification of Chinese Vehicle-2-X Message
ChineseV2XNetwork	CnV2xNet	Specification of Chinese Vehicle-2-X Network
ChineseV2XSecurity	CnV2xSec	Specification of Chinese Vehicle-2-X Security
CommunicationManagement	CM	Requirements on Communication Management; Specification of Communication Management
CommunicationStackTypes	Comtype	Specification of Communication Stack Types
Core	CORE	Specification of Adaptive Platform Core
CoreTest	CoreTst; CorTst	Requirements on Core Test; Specification of Core Test
CryptoDriver	Crypto	Specification of Crypto Driver
CryptoInterface	CryIf	Specification of Crypto Interface
CryptoServiceManager	Csm	Specification of Crypto Service Manager
CryptoStack	CryptoStack	Requirements on Crypto Stack
Cryptography	CRYPTO; CRYPT	Requirements on Cryptography; Specification of Cryptography
DDSCommunicationProtocol	DDS	Specification of DDS Service Communication Protocol
DDSSecurityIntegration	DDSS	Integration of DDS Security
DDSServiceDiscoveryProtocol	DDSSD	Specification of DDS Service Discovery Protocol
DIODriver	Dio	Requirements on DIO Driver; Specification of DIO Driver
DataDistributionService	Dds	Specification of Data Distribution Service for Classic Platform; Requirements on Data Distribution Service
DataDistributionServiceTransformer	DdsXf	Specification of Data Distribution Service Transformer
DebugTraceProfile	Arti; ARTIFO	Requirements on Debugging, Tracing and Profiling support of AUTOSAR Components
DefaultErrorTracer	Det	Specification of Default Error Tracer
DemandsConstraintsBaseSW	APBSW	Technical Report on Demands and Constraints on Base Software
Demonstrator	n/a	Source of Adaptive Platform Demonstrator

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
DemonstratorReleaseNotes	n/a	Adaptive Platform Demonstrator Release Notes
DiagnosticCommunicationManager	Dcm	Specification of Diagnostic Communication Manager
DiagnosticEventManager	Dem	Specification of Diagnostic Event Manager
DiagnosticExtractTemplate	DEXT	Diagnostic Extract Template
DiagnosticLogAndTrace	Dlt	Specification of Diagnostic Log and Trace
DiagnosticOverIP	DoIP	Specification of Diagnostic over IP
Diagnostics	DM; Diag	Specification of Diagnostics; Requirements on Diagnostics
DiagramSource	n/a	Explanation of Diagram Source
E2E	E2E	Requirements on E2E
E2ELibrary	E2E	Specification of SW-C End-to-End Communication Protection Library
E2EProtocol	E2E	E2E Protocol Specification
E2ETransformer	E2EXf	Specification of Module E2E Transformer
ECUConfiguration	ECUC; Cdd	Specification of ECU Configuration
ECUConfigurationParameters	n/a	Specification of ECU Configuration Parameters (XML)
ECUResourceTemplate	ECUR	Specification of ECU Resource Template
ECUStateManager	EcuM	Specification of ECU State Manager
EEPROMAbstraction	Ea	Specification of EEPROM Abstraction
EFXLibrary	Efx	Specification of Extended Fixed Point Library
ErrorDescription	ED	Description of the AUTOSAR standard errors
Ethernet	Eth	Requirements on Ethernet Support in AUTOSAR
EthernetDriver	Eth	Specification of Ethernet Driver
EthernetInterface	EthIf	Specification of Ethernet Interface
EthernetStateManager	EthSM	Specification of Ethernet State Manager
EthernetSwitchDriver	EthSwt	Specification of Ethernet Switch Driver
EthernetTransceiverDriver	EthTrcv	Specification of Ethernet Transceiver Driver
ExecutionManagement	EM	Requirements on Execution Management; Specification of Execution Management
FCDesignCommunicationManagement	n/a	Demonstrator Design of Functional Cluster Communication Management
FCDesignDiagnostics	n/a	Demonstrator Design of Functional Cluster Diagnostics
FCDesignExecutionManagement	n/a	Demonstrator Design of Functional Cluster Execution Management
FCDesignIdentityAndAccess Management	n/a	Demonstrator Design of Functional Cluster Identity and Access Management
FCDesignLogAndTrace	n/a	Demonstrator Design of Functional Cluster Log and Trace
FCDesignPersistency	n/a	Demonstrator Design of Functional Cluster Persistency
FCDesignPlatformHealthManagement	n/a	Demonstrator Design of Functional Cluster Platform Health Management
FCDesignStateManagement	n/a	Demonstrator Design of Functional Cluster State Management
FCDesignTimeSynchronization	n/a	Demonstrator Design of Functional Cluster Time Synchronization
FCDesignUpdateAndConfiguration Management	n/a	Demonstrator Design of Functional Cluster Update And Configuration Management

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
FeatureModelExchangeFormat	FMDT	AUTOSAR Feature Model Exchange Format
Features	FEAT	AUTOSAR Feature Model; Technical Report on AUTOSAR Features
Firewall	Fw	Specification of Firewall for Adaptive Platform; Specification of Firewall for Classic Platform; Requirements on Firewall
FirmwareOverTheAir	FOTA	Explanation of Firmware Over-The-Air; Requirements on Firmware Over-The-Air
FlashEEPROMEmulation	Fee	Specification of Flash EEPROM Emulation
FlashTest	FlsTst	Requirements on Flash Test; Specification of Flash Test
FlexRay	Fr	Requirements on FlexRay
FlexRayARTransportLayer	FrArTp	Specification of FlexRay AUTOSAR Transport Layer
FlexRayDriver	Fr	Specification of FlexRay Driver
FlexRayISOTransportLayer	FrTp	Specification of FlexRay ISO Transport Layer
FlexRayInterface	FrIf	Specification of FlexRay Interface
FlexRayNetworkManagement	FrNm	Specification of FlexRay Network Management
FlexRayStateManager	FrSM	Specification of FlexRay State Manager
FlexRayTransceiverDriver	FrTrcv	Specification of FlexRay Transceiver Driver
FunctionInhibitionManager	Fim	Requirements on Function Inhibition Manager; Specification of Function Inhibition Manager
FunctionalSafetyMeasures	n/a	Overview of Functional Safety Measures in AUTOSAR
GPTDriver	Gpt	Requirements on GPT Driver; Specification of GPT Driver
Gateway	GTW	Requirements on Gateway
General	AP	General Requirements specific to Adaptive Platform
GeneralBlueprints	n/a	Collection of blueprints for AUTOSAR Adaptive Platform M1 models; Collection of blueprints for AUTOSAR M1 models
GeneralBlueprintsSupplement	n/a	Supplementary material of general blueprints for AUTOSAR
GeneralDefinitions	n/a	Standardized M1 Models used for the Definition of AUTOSAR
GenericStructureTemplate	GST	Generic Structure Template
Glossary	n/a	Glossary
HWTestManagementIntegrationGuide	n/a	Specification and Integration of Hardware Test Management at start up and shutdown
HWTestManager	HTM; HTMSS	Requirements on HWTestManager; Requirements on Hardware Test Manager on start up and shutdown; Specification of Hardware Test Manager on start up and shutdown
HealthMonitoring	HM	Specification of Health Monitoring; Requirements on Health Monitoring
I2CDriver	I2C	Requirements on I2C Driver; Specification of I2C Driver
ICUDriver	Icu	Requirements on ICU Driver; Specification of ICU Driver
IEEE1722	IEEE1722	Requirements on IEEE1722

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
IEEE1722TransportLayer	IEEE1722Tp	Specification of IEEE1722 Transport Protocol Module
IFLLibrary	Ifl	Specification of Floating Point Interpolation Library
IFXLibrary	Ifx	Specification of Fixed Point Interpolation Library
IOHWAbstraction	IoHwAb	Requirements on I/O Hardware Abstraction
IOHardwareAbstraction	IoHwAb	Specification of I/O Hardware Abstraction
IPDUMultiplexer	IpduM	Requirements on I-PDU Multiplexer; Specification of I-PDU Multiplexer
IPsecImplementationGuidelines	n/a	Explanation of IPsec Implementation Guidelines
IPsecProtocol	IPSEC	Requirements on IPsec Protocol
ISO15118Charging	ISO15118Chrg	Specification of ISO15118 Charging
IdentityAndAccessManagement	n/a	Explanation of Identity and Access Management
InterfacesGuidelines	n/a	Guidelines for using Adaptive Platform interfaces
IntrusionDetectionSystem	Ids	Specification of Intrusion Detection System Protocol; Requirements on Intrusion Detection System
IntrusionDetectionSystemManager	AIDSM; IdsM	Specification of Intrusion Detection System Manager for Adaptive Platform; Specification of Intrusion Detection System Manager
KeyManager	KeyM	Specification of Key Manager
LIN	Lin	Requirements on LIN
LINDriver	Lin	Specification of LIN Driver
LINInterface	LinIf; LinTp	Specification of LIN Interface
LINStateManager	LinSM	Specification of LIN State Manager
LINTransceiverDriver	LinTrcv	Specification of LIN Transceiver Driver
LSduRouter	LSduR	Specification of Linklayer Sdu Routing Module
LanguageBindingForModeled APdatatypes	LBAP	Specification of Language Binding for modeled AP data types
LayeredSoftwareArchitecture	n/a	Layered Software Architecture
Libraries	LIBS	Requirements on Libraries
ListOfKnownIssuesSecureHardware Extensions	n/a	List of known Issues of Secure Hardware Extensions
LogAndTrace	LOG; LT	Specification of Log and Trace; Requirements on Log and Trace
LogAndTraceExtract	DLTXT	Log And Trace Extract Template
LogAndTraceProtocol	Dlt	Log and Trace Protocol Specification
MACsec	MACsec	Explanation of MACsec and MKA Protocols implementation and configuration guidelines; Requirements on MACsec
MACsecKeyAgreement	Mka	Specification of MACsec Key Agreement
MCUDriver	Mcu	Requirements on MCU Driver; Specification of MCU Driver
MFLLibrary	Mfl	Specification of Floating Point Math Library
MFXLibrary	Mfx	Specification of Fixed Point Math Library
MSFLibrary	Msf	Specification of MSFLibrary
MachineConfiguration	APMC	Adaptive Platform Machine Configuration
MachineConfigurationParameters	n/a	Specification of Machine Configuration Parameters

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
MacroEncapsulationofInterpolation Calls	n/a	Macro Encapsulation of Interpolation Calls
ManifestSpecification	MANI	Specification of Manifest
MemoryAbstractionInterface	MemIf	Specification of Memory Abstraction Interface
MemoryAccess	MemAcc	Specification of Memory Access
MemoryDriver	Mem	Specification of Memory Driver
MemoryHWAbstractionLayer	MemHwAb	Requirements on Memory Hardware Abstraction Layer
MemoryMapping	MemMap	Specification of Memory Mapping
MemoryServices	Mem	Requirements on Memory Services
MetaModel	n/a	Meta Model
Methodology	AMETH; METH	Methodology for Adaptive Platform; Methodology for Classic Platform
MiscSupport	n/a	AUTOSAR Miscellaneous Support Files
ModeManagement	ModeMgm	Requirements on Mode Management
ModeManagementGuide	MMG	Guide to Mode Management
ModelingShowCases	n/a	Modeling Show Cases Examples; Modeling Show Cases Report
NVDataHandling	n/a	NV Data Handling Guideline
NVRAMManager	NvM	Specification of NVRAM Manager
NetworkManagement	ANM; Nm	Specification of Network Management; Requirements on AUTOSAR Network Management
NetworkManagementInterface	Nm	Specification of Network Management Interface
NetworkManagementProtocol	Nm	Specification of the AUTOSAR Network Management Protocol
OCUDriver	Ocu	Requirements on OCU Driver; Specification of OCU Driver
OS	Os	Requirements on Operating System; Specification of Operating System
OperatingSystemInterface	OSI	Requirements on Operating System Interface; Specification of Operating System Interface
OperatingSystemTracingInterface	OSTI	Technical Report on Operating System Tracing Interface
PDURouter	PduR	Specification of PDU Router
PWMDriver	Pwm	Requirements on PWM Driver; Specification of PWM Driver
ParallelProcessingGuidelines	n/a	Design guidelines for using parallel processing technologies on Adaptive Platform
Persistency	PER	Requirements on Persistency; Specification of Persistency
PlatformDesign	n/a	Explanation of Adaptive Platform Design
PlatformHealthManagement	PHM	Requirements on Platform Health Management; Specification of Platform Health Management
PlatformTypes	APT; Platform	Specification of Platform Types for Adaptive Platform; Specification of Platform Types for Classic Platform
PortDriver	Port	Requirements on Port Driver; Specification of Port Driver
ProjectObjectives	PO	Project Objectives
QoSPoliciesInTheScopeOfSOMEIP	n/a	Explanation of QoS Policies in the scope of SOME/IP

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
RAMTest	RamTst	Requirements on RAM Test; Specification of RAM Test
RTE	Rte	Requirements on Runtime Environment; Specification of RTE Software
RawDataStream	RDS	Specification of Raw Data Stream
ReleaseOverview	n/a	Adaptive Platform Release Overview; Classic Platform Release Overview; Foundation Release Overview
RemotePersistency	RPER	Specification of Remote Persistency
SAEJ1939	J1939	Requirements on BSW Modules for SAE J1939
SAEJ1939DiagnosticCommunication Manager	J1939Dcm	Specification of a Diagnostic Communication Manager for SAE J1939
SAEJ1939FunctionalSafetyComm Protocol	J1939Fscp	Specification of a Functional Safety Communication Protocol Handler for SAE J1939
SAEJ1939NetworkManagement	J1939Nm	Specification of Network Management for SAE J1939
SAEJ1939RequestManager	J1939Rm	Specification of a Request Manager for SAE J1939
SAEJ1939TransportLayer	J1939Tp	Specification of a Transport Layer for SAE J1939
SOMEIPProtocol	SOMEIP	SOME/IP Protocol Specification; Requirements on SOME/IP Protocol
SOMEIPServiceDiscoveryProtocol	SOMEIPSD	SOME/IP Service Discovery Protocol Specification; Requirements on SOME/IP Service Discovery Protocol
SOMEIPTransformer	SomeIpXf	Specification of SOME/IP Transformer
SOMEIPTransportProtocol	SomeIpTp	Specification on SOME/IP Transport Protocol
SOVD	n/a	Explanation of Service-Oriented Vehicle Diagnostics
SPALGeneral	SPAL	General Requirements on SPAL
SPIHandlerDriver	Spi	Requirements on SPI Handler/Driver; Specification of SPI Handler/Driver
SWArchitecturalDecisions	n/a	Explanation of Adaptive and Classic Platform Software Architectural Decisions
SWArchitecture	n/a	Explanation of Adaptive Platform Software Architecture
SWCModeling	SWMG	Requirements on SW-C and System Modeling
SWCModelingGuide	SWMG; SWNR	SW-C and System Modeling Guide
SafeHardwareAcceleration	SHWA	Requirements on Safe Hardware Acceleration; Specification of Safe Hardware Acceleration
SafeHardwareAccelerationAPI	n/a	Explanation of Safe API for hardware accelerators
Safety	SAF	Safety Requirements for AUTOSAR Adaptive Platform and AUTOSAR Classic Platform
SafetyOverview	n/a	Explanation of Safety Overview
SafetyUseCase	n/a	Safety Use Case Example
SecOCProtocol	SecOc	Specification of Secure Onboard Communication Protocol
SecureHardwareExtensions	n/a	Specification of Secure Hardware Extensions
SecureOnboardCommunication	SecOC	Specification of Secure Onboard Communication; Requirements on Secure Onboard Communication
SecurityEventsSpecification	n/a	Technical Report on Security Events Specification
SecurityExtractTemplate	SECXT	Security Extract Template
SecurityOverview	n/a	Explanation of Security Overview

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
SensorInterfaces	ADI	Explanation of Sensor Interfaces; Specification of Sensor Interfaces
ServiceDiscovery	Sd	Specification of Service Discovery
SocketAdaptor	SoAd	Specification of Socket Adaptor
SoftwareClusterConnection	SwCluC	Requirements on Software Cluster Connection module; Specification of Software Cluster Connection module
SoftwareComponentTemplate	SWCT	Software Component Template
SpecificationsARXML	n/a	Specifications in ARXML format
StandardTypes	Std	Specification of Standard Types
StandardizationTemplate	STDT	Standardization Template
StateManagement	SM	Requirements of State Management; Specification of State Management
SwClusterDesignAndIntegration Guideline	n/a	Explanation of Software Cluster Design And Integration Guideline for Classic Platform
SynchronizedTimeBaseManager	StbM	Specification of Synchronized Time-Base Manager
SystemHealthMonitoring	n/a	Explanation of System Health Monitoring
SystemTemplate	SYST	System Template
SystemTests	n/a	System Tests for Adaptive Platform Demonstrator
TcpIp	TcpIp; IKE	Specification of TCP/IP Stack
TimeSensitiveNetworkFeatures	n/a	Explanation of Time Sensitive Network features
TimeService	Tm	Requirements on Time Service; Specification of Time Service
TimeSync	TS	Requirements on Time Synchronization
TimeSyncOverCAN	CanTSyn	Specification of Time Synchronization over CAN
TimeSyncOverCANProtocol	CanTSyn	Time Synchronization over CAN Protocol Specification
TimeSyncOverEthernet	EthTSyn	Specification of Time Synchronization over Ethernet
TimeSyncOverEthernetProtocol	TS	Time Synchronization over Ethernet Protocol Specification
TimeSyncOverFlexRay	FrTSyn	Specification of Time Synchronization over FlexRay
TimeSynchronization	TS	Specification of Time Synchronization
TimingAnalysis	n/a	Timing Analysis and Design
TimingExtensions	TIMEX	Specification of Timing Extensions for Adaptive Platform; Specification of Timing Extensions for Classic Platform
Transformer	Xfrm	Requirements on Transformer
TransformerGeneral	Xfrm	General Specification of Transformers
UDPNetworkManagement	UdpNm	Specification of UDP Network Management
UpdateAndConfigurationManagement	UCM	Requirements on Update and Configuration Management; Specification of Update and Configuration Management
UtilizationOfCryptoServices	n/a	Utilization of Crypto Services
V2XBasicTransport	V2xBtp	Specification of Vehicle-2-X Basic Transport
V2XCommunication	V2X	Requirements on Vehicle-2-X Communication
V2XDataManager	V2xDM	Specification of Vehicle-2-X Data Manager
V2XFacilities	V2xFac	Specification of Vehicle-2-X Facilities
V2XGeoNetworking	V2xGn	Specification of Vehicle-2-X Geo Networking
V2XManagement	V2xM	Specification of Vehicle-2-X Management
V2XRemoteAccessLayer	V2xRAL	Vehicle-2-X Remote Access Layer Protocol Specification

▽

△

Document name (full)	*Document name (abbreviation)*	*Document title*
VDP	VDP	Vehicle Data Protocol Specification; Requirements on Vehicle Data Protocol
VDPCMRemote	VdpCmR	Specification of VDP Communication Module Remote
VFB	VFB	Virtual Functional Bus
VSSRepresentation	VSS	Technical Report on VSS Representation
VehicleUpdateAndConfiguration Management	VUCM	Requirements on Vehicle Update and Configuration Management; Specification of Vehicle Update and Configuration Management
WatchdogDriver	Wdg	Requirements on Watchdog Driver; Specification of Watchdog Driver
WatchdogInterface	WdgIf	Specification of Watchdog Interface
WatchdogManager	WdgM	Specification of Watchdog Manager
WirelessEthernetDriver	WEth	Specification of Wireless Ethernet Driver
WirelessEthernetTransceiverDriver	WEthTrcv	Specification of Wireless Ethernet Transceiver Driver
WorkflowExample	n/a	Technical Report on Supplementary Material of Workflow Example
XCP	Xcp	Requirements on Module XCP; Specification of Module XCP
XMLSchema	n/a	Meta Model-generated XML Schema
XMLSchemaProductionRules	XMLSPR	AUTOSAR XML Schema Production Rules
XMLSchemaSupplement	n/a	Supplementary material of the AUTOSAR XML Schema

⌋

2.5 Standardized naming of published files

[TPS_STDT_00050] Standardized naming convention for published AUTOSAR files ⌈Files publicly released by AUTOSAR shall have the naming convention: AUTOSAR_{doc-std-abbr}_{doc-cat-abbr}_{doc-name-full}⌋

2.6 Standardized ARPackage.shortNames

Selected M1 ARXML models published by AUTOSAR use well-known ARPackage.shortNames. As stated in [TPS_GST_00080] and [TPS_GST_00081], the top-level ARPackage=AUTOSAR is fixed, i.e. '/AUTOSAR'. Further, directly under the top-level ARPackage, the next level ARPackage has also reserved a subset of ARPackage.shortNames for specific usages. These are listed in [TPS_STDT_00149].

[TPS_STDT_00149] AUTOSAR reserved shortNames under the top-level ARPackage=AUTOSAR ⌈

shortName	Usage	Context
Documents	Common namespace of AUTOSAR documents	-
AISpecification	Specification of application interfaces	[3, MOD-AISpecification]
Features	ARXML representation AUTOSAR feature graph from [4]	[5, MOD-Features]
Diagnostic	Definition of base types used in diagnostic context	[6, MOD-GeneralBlueprints]
GenDef	General standardized ARXML definitions	[7, MOD-GeneralDefinitions]
NvBlockSoftwareComponentType	Blueprint of ClientServerInterface between a ApplicationSwComponentType and a NvBlockSwComponentType	[6, MOD-GeneralBlueprints]
TestCases	Blueprint test scenarios to validate AUTOSAR delivered models	[6, MOD-GeneralBlueprints]
StdTypes	AUTOSAR Adaptive Platform standardized StdCppImplementationDataTypes	[7, MOD-GeneralDefinitions]
MachineFunctionGroupStates	AUTOSAR Adaptive Platform standardized values for machine function group states	[7, MOD-GeneralDefinitions]

⌋

3 Tracing

3.1 AUTOSAR document traceable items

AUTOSAR documents utilize a fixed notation to emphasize certain specification texts of importance to the user. Known in AUTOSAR as traceable items and depending on the characteristics thereof, they may be decorated with certain other meta-data attributes. In general, traceables:

- maintain a unique identifier attribute
- maintain a lifecycle state attribute
- contain a body of specification text between an "Opening Half Bracket" (Unicode: Left Ceiling {0x2308}) and a "Closing Half Bracket" (Unicode: Right Floor {0x230B})
- shall be tracked for changes over each release

These types of traceables, the general attributes of traceables and further traceable-specific attributes are explained in the next sub-chapters.

3.1.1 Specification Item

[TPS_STDT_00080] Representation of specification items in AUTOSAR documents ⌈AUTOSAR specification items are represented using the structure with the following attributes:

- The headline consists of an Id (shortName) which shall be written inside squared brackets and shall follow [TPS_STDT_00042].
- After the Id the LifeCycleState follows in curly brackets. The allowed values are VALID, DRAFT and OBSOLETE and shall follow [TPS_GST_00051]. If there is no LifeCycleState information stated then the state is VALID.
- After the LifeCycleState an optional specification item title (longName) should be stated to improve human readability.
- The next line starts with an opening half bracket and the content of the specification item follows. The end of it shall be marked by the closing half bracket.
- After the closing half bracket an opening round bracket indicates the comma separated list of requirements which are fulfilled by this specification item. The end of it shall be marked by the closing round bracket. If no up traces are available the round brackets shall be written with empty content.
- The specification items shall describe the semantics and syntax of models.

⌋

[TPS_STDT_00042] Naming convention for SPECIFICATION_ITEMs ⌈AUTOSAR SPECIFICATION_ITEMs shall follow the naming convention:

```
[{doc-std-abbr}_] {doc-cat-abbr} _ {doc-name-abbr} _ [{special}_]
{num}
```

where:

- `[]`: optional
- `num`: unique number (00000 .. 99999)
- `special`: `specialization`, one of:
  - "CONSTR" as per [TPS_STDT_00089]
  - "NA" as per [TPS_STDT_00056]

⌋

3.1.2 Model Constraint

[TPS_STDT_00081] Representation of constraint items in AUTOSAR template documents ⌈AUTOSAR constraint items in template documents are represented using the structure with the following attributes:

- The constraint Identifier `shortName` is comprised of:
  - a string prefix: "constr_"
  - a numerical suffix: 4-5 digits long
  - the leading digit of the numerical suffix shall be ≥ 1

  On documentation level, this shall be rendered inside [] brackets. The numerical suffix is globally unique.
- After the Id the `LifeCycleState` follows in curly brackets. The allowed values are VALID, DRAFT and OBSOLETE and shall follow [TPS_GST_00051]. If there is no `LifeCycleState` information stated then the state is VALID.
- After the `LifeCycleState` the constraint title (`longName`) follows.
- The constraint content shall be written inside the opening and closing half bracket.
- The constraint items shall further restrict the validity of models.
- An optional constraint `ImpositionTime` ([TPS_STDT_00095])

⌋

3.1.3 Software Constraint

[TPS_STDT_00088] Representation of constraint items in AUTOSAR non template documents ⌈AUTOSAR constraint items in AUTOSAR non template documents are represented using the structure with the following attributes:

- The headline consists of an Id (`shortName`) which shall be written inside squared brackets and shall follow [TPS_STDT_00042].
- After the Id the `LifeCycleState` follows in curly brackets. The allowed values are VALID, DRAFT and OBSOLETE and shall follow [TPS_GST_00051]. If there is no `LifeCycleState` information stated then the state is VALID.
- After the `LifeCycleState` the constraint title (`longName`) follows.
- The constraint content shall be written inside the opening and closing half bracket.
- This Id is independent of a `non-specialized` Id, i.e. `SWS_XXX_12345` != `SWS_XXX_CONSTR_12345`

⌋

In a constraint according to [TPS_STDT_00088], the term “shall” shall be used inside the content to underline the mandatory intention of the item.

[TPS_STDT_00089] `SPECIFICATION_ITEM`s with constraint semantics ⌈`SPECIFICATION_ITEM`s which express a constraint semantic may use the infix special="CONSTR" (see [TPS_STDT_00042]). This infix is permitted only in an ASWS or SWS, in other document types this is forbidden⌋

[TPS_STDT_00148] `SPECIFICATION_ITEM`s which are not-applicable for up-tracing ⌈`SPECIFICATION_ITEM`s which shall be excluded (not applicable) from up-tracing shall use the infix special="NA" (see [TPS_STDT_00042]). This infix is permitted only in an SWS or PRS, in other document types this is forbidden⌋

3.1.4 Model Advisory

[TPS_STDT_00093] Representation of advisory items in AUTOSAR template documents ⌈AUTOSAR advisory items in template documents are represented with the following format:

- The advisory Identifier `shortName` is comprised of:
  - a string prefix: "advisory_"
  - a numerical suffix: 4-5 digits long
  - the leading digit of the numerical suffix shall be ≥ 1

On documentation level, this shall be rendered inside [] brackets. The numerical suffix is globally unique.

- After the Identifier the `LifeCycleState` follows in curly brackets. The allowed values are VALID, DRAFT and OBSOLETE and shall follow [TPS_GST_00051]. If there is no `LifeCycleState` information stated then the state is VALID.
- After the `LifeCycleState` the advisory title `longName` follows.
- The advisory content shall be written inside the opening (ceil) and closing (floor) symbols.

⌋

In an advisory item according to [TPS_STDT_00093], the term “should” shall be used inside the content to underline the advisory intention of the item.

3.1.5 Imposition Time of a Model Constraint/Model Advisory

The timing of when precisely an AUTOSAR tool should enforce a model constraint/advisory is in some cases clear and in other cases conditional. For that reason AUTOSAR constraints/advisories may contain an additional `ImpositionTime` attribute to stipulate the latest point in time in a workflow when the constraint/advisory shall be applied[1]. Since AUTOSAR defines separate Methodology workflows for the Classic Platform and Adaptive Platform, the respective `ImpositionTime`s are in general also platform dependent.

[TPS_STDT_00095] Semantics of an `ImpositionTime`

Status: DRAFT

⌈An AUTOSAR model constraint/advisory contains an `ImpositionTime`, which specifies the latest point in the methodology workflow when the respective constraint/advisory shall be imposed on a model.⌋

[TPS_STDT_00096] Application of an `ImpositionTime`

Status: DRAFT

⌈A constraint/advisory may be enforced before the `ImpositionTime` but shall be enforced no later than the respective point in the workflow.⌋

[TPS_STDT_00097] Semantics of an unspecified `ImpositionTime`

Status: DRAFT

⌈An AUTOSAR model constraint/advisory with no `ImpositionTime`, implies "at an arbitrary point in the workflow".⌋

[1] Typically by an authoring/modeling tool

Imposition Time	*Description*	*Motivation*
IT_RteGen	RTE is generated	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. *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. In other words, a constraint that is imposed *at the time when the contract phase generation is executed* shall also be imposed *at the time when the RTE is generated*.
IT_CpgExe	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.
IT_CompSwcT	Creation of the CompositionSw ComponentType is finished	This imposition time applies to the creation of compositions of software-components. This imposition time is considered optional. In other words, there may be use cases to deliver `CompositionSwComponentType`s that violate constraints with this imposition time to another party. But it may also make sense in some cases to make sure, that a `CompositionSwComponentType` that is going to be delivered to another party fulfills the constraints associated with this binding time.
IT_Apsd	Creation of the Abstract Platform System Description is finished	This imposition time indicates when the Abstract Platform System Description is complete.
IT_SysDesc	SYSTEM_DESCRIPTION is completed	This imposition time is aimed at the time when a system description (e.g. SYSTEM_DESCRIPTION or SYSTEM_ EXTRACT or ECU_SYSTEM_DESCRIPTION) is complete for exchange between parties and is ready as input for Ecu Extract and Com Stack generators.
IT_SwCluSysDesc	SW_CLUSTER_SYSTEM_ DESCRIPTION is completed	This imposition time denotes the step in the workflow where the SW_CLUSTER_SYSTEM_DESCRIPTION model is considered complete such that the development and integration of the Software Cluster can start.
IT_EcuExt	ECU_EXTRACT is completed	This imposition time denotes the step in the workflow where the ECU_EXTRACT model is considered complete such that it can be used as input for the generation of the RTE.
IT_ResPool	Definition of the resource pool is finished	This imposition time denotes the step in the workflow where the pool of resources which can be provided or required by Software Clusters is considered complete such that such that the development and integration of the Software Cluster can start.
IT_VfbTd	VFB Timing Description is completed	This imposition time is aimed at the time when a VFB Timing is complete.
IT_SwcTd	Swc Timing Description is completed	This imposition time is aimed at the time when a Swc Timing is complete.
IT_SysTd	System Timing Description is completed	This imposition time is aimed at the time when a System Timing is complete.
IT_BswTd	Bsw Timing Description is completed	This imposition time is aimed at the time when a Bsw Timing is complete. This applies for both the Bsw Module Timing and the Bsw Composition Timing.
IT_EcuTd	Ecu Timing Description is completed	This imposition time is aimed at the time when a Ecu-wide Timing is complete.

△

Imposition Time	*Description*	*Motivation*
IT_SubClasTdEvAss	Imposition time associated with the concrete subclass of Timing DescriptionEvent is applied.	This means that the imposition time of the constraint cannot be unambiguously defined on the level of the abstract meta-class `TimingDescriptionEvent`. Sub-classes of `TimingDescriptionEvent` have imposition times associated with them (by means of constraints that refer to the subclasses) and the constraints that apply in the context of the definition of `TimingDescriptionEvent` shall therefore not contain a concrete imposition time but take over the imposition time from the applicable subclass. Example: subclass `TDEventVfb` is associated with the imposition time at the time when the VFB Timing Description is complete.
IT_SubClasTeAss	Imposition time associated with the concrete subclass of Timing Extension.	This means that the imposition time is relative to the concrete subclass of TimingExtension (Timing View) in use, namely: - at the time when the VFB Timing Description is complete, - at the time when the Swc Timing Description is complete, - at the time when the System Timing Description is complete, - at the time when the Bsw Timing Description is complete, - at the time when the Ecu Timing Description is complete
IT_SubClasTdEv	Imposition time associated with the concrete subclass of Timing DescriptionEvent by condition.	The imposition time is associated with the concrete subclass of `TimingDescriptionEvent` if the constraint is applied to a `TimingDescriptionEvent` or at the imposition time associated with the concrete subclass of TimingExtension if the constraint is applied to a `TimingDescriptionEventChain`.
IT_BefAraApiGen	Before the generation of the ara API starts	This imposition time is aimed at the time when a software-component is ready for generating the header files such that the implementation of the software-component can be started.
IT_DesExe	Design of the Executable is completed	This imposition time is aimed at the time when an `Executable` is finished, i.e. it shall be used in constraints that target the consistency of the modeling of `Executable`.
IT_ProDes	ProcessDesign is completed	This imposition time is aimed at the time when a `ProcessDesign` is finished, i.e. it shall be used in constraints that target the consistency of the modeling of `ProcessDesign`.
IT_GraDes	GrantDesign is completed	This imposition time is aimed at the time when a `GrantDesign` is finished, i.e. it shall be used in constraints that target the consistency of the modeling of `GrantDesign`.
IT_SysDes	System design is completed	This imposition time denotes the step in the workflow, where the system design is about to be finished.
IT_SubSysDes	Sub-system design is completed	This imposition time denotes the step in the workflow, where the sub-system design is about to be finished.
IT_DiagDes	Diagnostic design is completed	This imposition time denotes the step in the workflow, where the diagnostic design is about to be finished independent of the standard where it is applied. This includes the finalization, e.g., of the DEXT.
IT_MachDes	Machine design is completed	This imposition time denotes the step in the workflow, where the machine design is about to be finished.
IT_Mani	Creation of the manifest is finished	This imposition time denotes the step in the workflow, where the manifest is considered complete such that the installation on a target platform can be started.
IT_FeatMod	Feature Model is completed	This imposition time denotes the step in the workflow, where the feature model is about to be finished.
IT_LogTrace	Log and Trace Extract is complete	This imposition time denotes the step in the workflow, where the Log and Trace Extract is about to be finished.

▽

△

Imposition Time	*Description*	*Motivation*
IT_ValSpec	ValueSpecification is applied	This imposition time is aimed at the point in time where a `ValueSpecification` is applied to data object and consistency requirements between the `ValueSpecification` and the data object can be checked.
IT_BinObjMetaData	The definition of binary object meta-data is finished	This imposition time denotes the step in the workflow where the description of `CpSoftwareClusterBinaryManifestDescriptor` is considered complete so that that the Software Cluster Binary Manifest can be defined during the integration.
IT_BswMD	Configuration of the BSW module is finished	This imposition time is applicable at the time when the BSW module description is complete.
IT_CfgFc	Configuration of Functional Cluster is finished	This imposition time denotes the step in the workflow, where the configuration of a functional cluster is considered complete such that the installation on a target platform can be started.

Table 3.1: Overview Imposition Times

3.1.6 Requirement

[TPS_STDT_00078] Representation of requirements in AUTOSAR documents ⌈AUTOSAR requirements are represented using the structure of [TPS_STDT_00060] where the following attributes are presented as a table:

- The headline shall contain the Id (`shortName`), the `LifeCycleState` (type) and a unique short text (`longName`) of the requirement.
- The value of Type shall be one of "valid", "draft" or "obsolete", see [TPS_STDT_00064].
- The description of requirement contains of a complete English sentence using the sentence pattern [TPS_STDT_00094] including one of the keywords from [TPS_STDT_00053]. Additional information: needed to understand the requirement, can be added to the description.
- The rationale can be used to justify or rationalize the requirement.
- Use case can be used to describe the use case of the requirement.
- Applies to shall contain a comma separated tag list with one of the following values from `StandardNameEnum`.
- Dependencies may contain references to other requirements in this document which this requirement depends on.
- Supporting material can be used for documenting references to other documents or models that support the implementation of this requirement.

⌋

[TPS_STDT_00056] Representation of not-applicable requirements in AUTOSAR documents ⌈For those requirements which are not-applicable to a particular specification, [TPS_STDT_00042] allows the `special` to be `NA`.

In order to apply this, specification item with the `shortName` e.g (`[RS_XXX_NA]` or even `[RS_XXX_NA_00099]`) may be created which traces back to the not-applicable requirement items.

By this, not-applicable requirements are easily identified in requirements tracing tables. Requirements tracing is complete since it also explicitly expresses the not-applicable requirements.⌋

Notes to [TPS_STDT_00056]:

- Not-applicable requirements shall always trace up to an `RS`
- Not-applicable requirements shall never trace up to a `RS_XXX_NA_*` requirement

3.1.6.1 Phrasing convention

In case no data is available for a dedicated field in a requirement table, it may be empty. The description of a requirement follows a dedicated sentence pattern.

$< OptionalCondition >$ –> $< Subject >$ –> $< Shall >$ –> $< Statement >$

[TPS_STDT_00094] Sentence pattern ⌈The sentence pattern is built up by:

- $< OptionalCondition >$: A condition under which the $< Statement >$ shall be true. The condition starts with either `if` or `when` and ends with then. Where `when` identifies an event. I.e. the point in time when the condition becomes true. In natural language you could use "as soon as" to express the same. `If` in contrast identifies a static condition which is independent from time. For static conditions you may add `else` (optionally) after the $< Statement >$ to express an alternative requirement by appending it as an additional sentence following the pattern.
- $< Subject >$: The item that is to fulfill the $< Statement >$. Remark: The subject typically represents your subject under development, a property or a part of it. It is highly recommended to maintain an overview of the subjects you are specifying in the introductory section of your specification document.
- shall: Separates the $< Subject >$ from the $< Statement >$ and identifies (partial) requirements.
- $< Statement >$: A statement that can either be verified or falsified. If the statement is true, then the $< Subject >$ satisfies the requirement, otherwise it does not.

⌋

[TPS_STDT_00053] Expression of obligation ⌈The following verbal forms for the expression of obligation shall be used to indicate requirements.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as follows.

Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the adjective "LEGALLY REQUIRED", means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase "MUST NOT", means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

An implementation, which does not include a particular option, SHALL be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)⌋

3.1.7 Applicability of Requirements

AUTOSAR `Requirement`s may be written such that they may, or may not, be implementable in a single platform or cross-platform.

- If the placement of a Requirement is in AUTOSAR platform=CP or platform=AP the scope is "known" to be restricted to the respective platform
- If the placement of a Requirement is in AUTOSAR platform=FO, it is ambiguous which platform it is applicable to. It must therefore be decorated with the appliesTo attribute to explicitly state the platform scope of the Requirement

[constr_2603] Use of appliesTo in context of the specification level ⌈On specification level 1 and 2 only the requirements table including the appliesTo attribute shall be used. On the specification levels 3 and 4 only the requirements table without the appliesTo shall be used. Exception: Documents of the foundation which are handled on specification level 3.⌋

Rational: This avoids unintentional cross references which disturb the structure of tracing.

[constr_2604] Allowed up-traces in context of appliesTo values ⌈Traces to documents of upper specification levels shall be conform to the values assigned to appliesTo.⌋

Note: Optional requirements on level 1 to 4 of the AUTOSAR requirements hierarchy are not allowed. An optional part of an implementation is only optional for the end-user of AUTOSAR. In order to provide this option, the corresponding choice shall be mandatory in the according specification. That means, a feature described as "AUTOSAR should support X" can never be correct, because the underlying requirements layer is always static and would have no chance to decide whether "X" should be part of it or not. A correct writing would be e. g. "AUTOSAR shall support optional X".

3.1.8 Meta-classes supporting Traceable Items

[TPS_STDT_00001] Support bottom up tracing ⌈Standardization Template supports bottom up tracing between these levels by the meta-class Traceable. This allows to represent traceable entities and to establish traces between those. These entities reside within a DocumentationBlock. One prominent place is DocumentationBlock.trace in particular within Identifiable.introduction.⌋

The abstract class Traceable is specialized, targeting differing usages in the next sections.

[constr_2565] Traceable shall not be nested ⌈Due to the intended atomicity of requirements respectively specification items, Traceable shall not be nested.⌋

3.1.8.1 StructuredReq

[TPS_STDT_00060] **StructuredReq** ⌈This represents a structured requirement as it is used within AUTOSAR RS documents.⌋

3.1.8.2 TraceableText/TraceableTable

[TPS_STDT_00098] Standardized categorys of TraceableText and TraceableTable ⌈

- CONSTRAINT_ITEM: represents a Traceable with constraint semantics. It is similar to a specification item but represents issues that may be validated automatically e.g. by a tool.
- ADVISORY_ITEM: represents a Traceable with advisory semantics. It is similar to a constraint item but represents the characteristic of a WARNING rather than an ERROR.
- REQUIREMENT_ITEM: represents a requirement in a requirement specification.
- SPECIFICATION_ITEM: represents an AUTOSAR item of specification. Such an item is a requirement for the implementation of the software specification.

⌋

[TPS_STDT_00052] Characteristics of TraceableText ⌈TraceableText should[2] be:

- **identifiable**: TraceableText shall be identified by a unique shortName (see [TPS_STDT_00042]). This is automatically fulfilled by applying the AUTOSAR meta-model and schema.
- **specific**: TraceableText should be written such that the content is unambiguous and comprehensive - even if this would not result in an elegant writing style.
- **atomic**: One TraceableText should cover one particular issue.
- **verifiable**: The content of TraceableText should be written concrete such that it can be verified - not necessarily automatically but at least by human experts.

  In particular the requirement levels specified in [TPS_STDT_00053] shall be applied.

⌋

[TPS_STDT_00054] Organisation of TraceableText ⌈A set of TraceableText within a specification shall have the following properties:

[2]This usage of the word "should" indicates that this is not always easy to decide. For example [TPS_STDT_00052] could also have been divided in one TraceableText per item.

- **hierarchical structure:** Multiple `TraceableText`s shall be structured in several successive levels - this is mostly ensured by the templates for the different kind of AUTOSAR specifications.
- **completeness:** `TraceableText` at one level shall fully implement all `TraceableText` of the previous level.
- **external consistency:** Multiple `TraceableText`s shall not contradict each other.
- **no duplication of information within any level of the hierarchical structure**: The content of one `TraceableText` shall not be repeated in any other `TraceableText` within the same level of the hierarchical structure.
- **maintainability:** A set of `TraceableText` can be modified or extended, e.g. by introduction of new versions of TraceableText or by adding/removing `TraceableText`. The `shortName` of `TraceableText` shall not be reused or changed.

⌋

3.2 Trace levels

AUTOSAR permits that `SPECIFICATION_ITEM`s and `Requirement`s may trace upwards. A `SPECIFICATION_ITEM` may up-trace to a `Requirement` and a (fine-grained) `Requirement` may up-trace to (coarse-grained) `Requirement`. This is shown in as shown in 3.1. Identifiers of `traceable items` are explained in [TPS_STDT_00042].

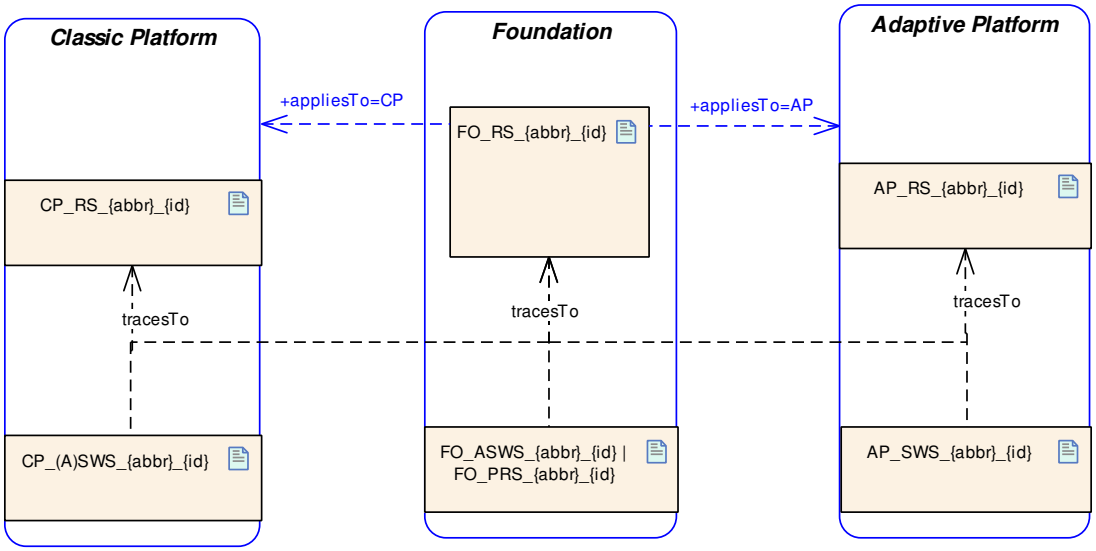


Figure 3.1: AUTOSAR `traceable items` up-tracing

Notes:

- TPS documents: up-tracing is not permitted
- TR, EXP documents: up-tracing is optional

3.3 Trace groups

AUTOSAR supports the logical assignment of Requirements into groups, either on the level of:

- a singular StructuredReq
- all StructuredReqs in a document chapter

to one or more Trace Groups (TGs).

[TPS_STDT_00099] Standardized naming convention for trace groups ⌈Trace groups shall follow the naming convention: TG_<rs-abbr>_<name> where:

- <rs-abbr> = is an abbrName from [TPS_STDT_00137]
- <name> = used to further sub-divide the trace group, e.g. if the trace group targets a specific SWS it can be derived from <rs-abbr>, with the following caveats:
  - <name>="Functional" is forbidden
  - <name>="NonFunctional" needs no further refinement (no traces from SWS/PRS/ASWS to such requirements)

⌋

Examples:

- TG_AP_NonFunctional: Trace group of "non-functional" requirements in [8, AP-RS-General]. Note: In this instance the "AP" part of the name corresponds to the <rs-abbr> of [8])
- TG_BSW: Trace group of all requirements in [9, CP-RS-BSWGeneral]
- TG_BSW_NonFunctional: Trace group of all non-functional requirements in [9, CP-RS-BSWGeneral]
- TG_J1939_Tp: Trace group of those requirements in [10, CP-RS-SAEJ1939] - (<rs-abbr>="J1939") which have a relation to the [11, CP-SWS-SAEJ1939TransportLayer]
- TG_J1939_Rm: Trace group of those requirements in [10, CP-RS-SAEJ1939] - (<rs-abbr>="J1939") which have a relation to the [12, CP-SWS-SAEJ1939RequestManager]
- TG_J1939_NonFunctional: Trace group of non-functional requirements in [10, CP-RS-SAEJ1939] - (<rs-abbr>="J1939")

4 Life Cycle of AUTOSAR definitions

In order to support evolution and backward compatibility of the standardized model elements like port prototype blueprints, port interfaces, keyword abbreviations, SW-Cs (in ASW) or of the API of a BSW module etc., AUTOSAR supports life cycles. The meta-model and the details of the application of this meta-model is specified in chapter "Life Cycle Support" of [2].

[TPS_STDT_00038] Life Cycle Support ⌈The Standardization template is able to express information about the state of the blueprints by references from within a `LifeCycleInfoSet`.⌋

[TPS_STDT_00064] Applied Life Cycle Information Sets on AUTOSAR provided Models (M1) ⌈The following `LifeCycleState`s are applied for AUTOSAR provided model elements:

- `VALID`: This indicates that the related entity is a valid part of the document. This is the default.
- `DRAFT`: This indicates that the related entity is introduced newly in the model but still experimental. This information is published but is subject to be changed without backward compatibility management.
- `OBSOLETE`: This indicates that the related entity is obsolete and kept in the model for compatibility reasons. If this tag is set, the note shall express the recommended alternative solution.
- `REMOVED`: This indicates that the related entity is removed from the model. It shall not be used and should not even appear in documents. An AUTOSAR release does not contain such elements. It is intended for AUTOSAR internal development.

  Even if such removed elements are not included in an `.arxml` they can still be referenced in a `LifeCycleInfoSet` by using the ≪`atpUriDef`≫ attribute of type `Referrable`: `lcObject`, respectively `useInstead`.

If an object is not referenced in a `LifeCycleInfoSet`, the related entity is a valid part of the current model.⌋

Note that according to [TPS_STDT_00064] if there is no life cycle information for an element then it is defined that the element is valid. In other words, in general there is no need to define a `LifeCycleInfoSet` with `defaultLcState`=`VALID`. Nevertheless, there might be use cases when it could be useful to explicitly define such a `LifeCycleInfoSet`. For example if element "x" gets `LifeCycleState`=`OBSOLETE` and subsequently this is identified as an error and the life cycle returns back to `VALID`. This could be documented in such a `LifeCycleInfoSet`.

An ARXML representation of the life cycle according is provided with [TPS_GST_-00051].

4.1 Life Cycle State vs Tracing Levels

[constr_2625] Permitted `LifeCycleState` combinations in a requirement uptrace ⌈

	Trace to: TraceableText.category=REQUIREMENT_ITEM			
Trace from:	DRAFT	VALID	OBSOLETE	REMOVED
DRAFT	1	1		
VALID	x	1		
OBSOLETE	1	1	1	
REMOVED	1	1	1	1
Legend:				
x) A "not applicable" requirement - as per [TPS_STDT_00056] with LifeCycleState==VALID may uptrace to LifeCycleState==DRAFT 1) Permitted				

⌋

5 Blueprints

5.1 The Principles of Blueprints

[TPS_STDT_00002] The Principles of Blueprints ⌈This chapter describes the support of the AUTOSAR meta-model for the pre-definition of model elements taken as the basis for further modeling. These pre-definitions are called blueprints.⌋

For example, an authoring tool provides the such predefined `PortInterface` as a kind of toolbox from which the definitions can be copied to a project.

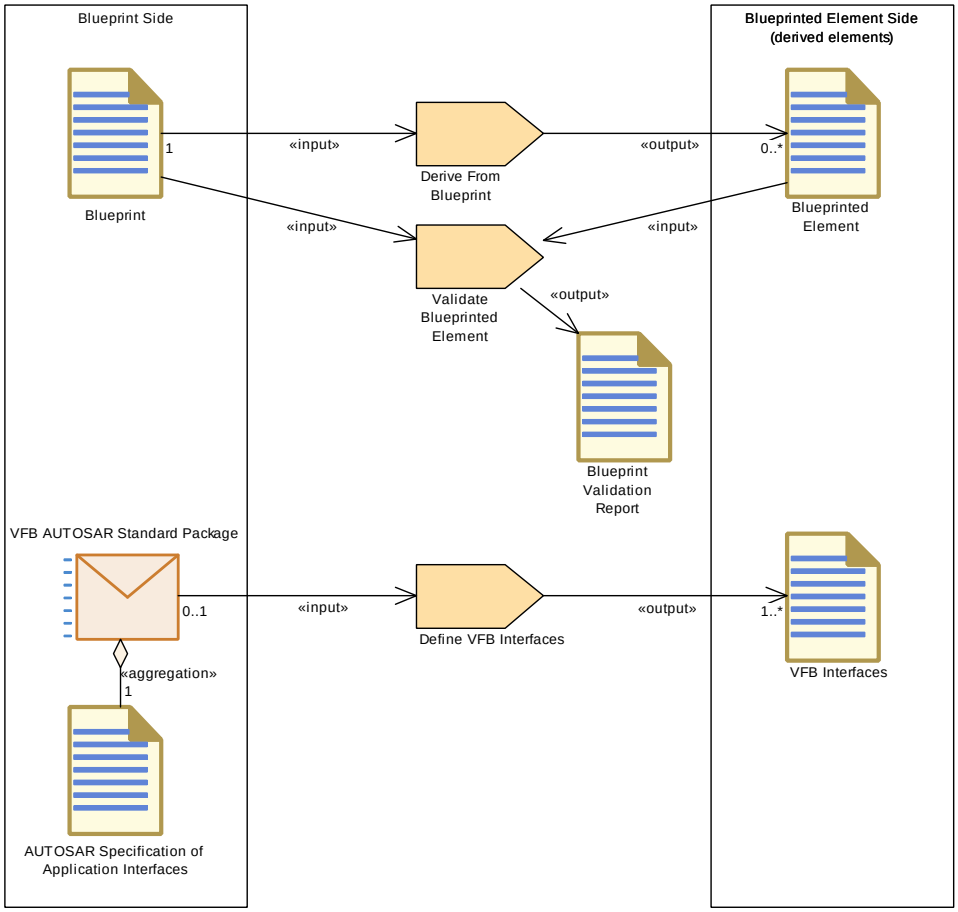


Figure 5.1: Blueprint methodology approach

Figure 5.1 illustrates the use case. The blueprint is on one hand used as an input to derive objects (DeriveFromBlueprint) and later also used to validate the derived objects. As an example the figure shows that the application interfaces are used to derive VFB interfaces (namely `PortInterface`s).

5.1.1 Abstract pattern for Blueprints

The blueprint approach is represented by the abstract blueprint structure as shown in figure 5.2. It is based on three entities:

- **Blueprint**, represented by `AtpBlueprint`, acts as the pre-definition of the element. Basically it follows the same structure as the derived elements.

  But there might be additional elements to support the fact that it is a blueprint. An example for this is that `PortPrototypeBlueprint` also specifies `initValue`s which is not the case for `PortPrototype` which get their initial values from appropriate `ComSpec`s.

- **Blueprinted Element**, represented by `AtpBlueprintable`, acts as the element which was derived from the Blueprint. These elements are derived from blueprints mainly by copy and refine. This "refine" may add further attribute values, update `shortName` etc. The details of possible refinements are specified for each blueprint individually.

  Note that the subsequent processing of blueprinted elements (e.g. RTE generation) does not refer to the blueprints anymore.

- **Blueprint Mapping**, represented by `AtpBlueprintMapping`, acts as a reference between blueprints and their derived elements. The main purpose of this blueprint mapping is to

  – provide the ability to validate for each derived element that they conform to the blueprint.

  – reflect the fact that the derived elements are part of a common concept.

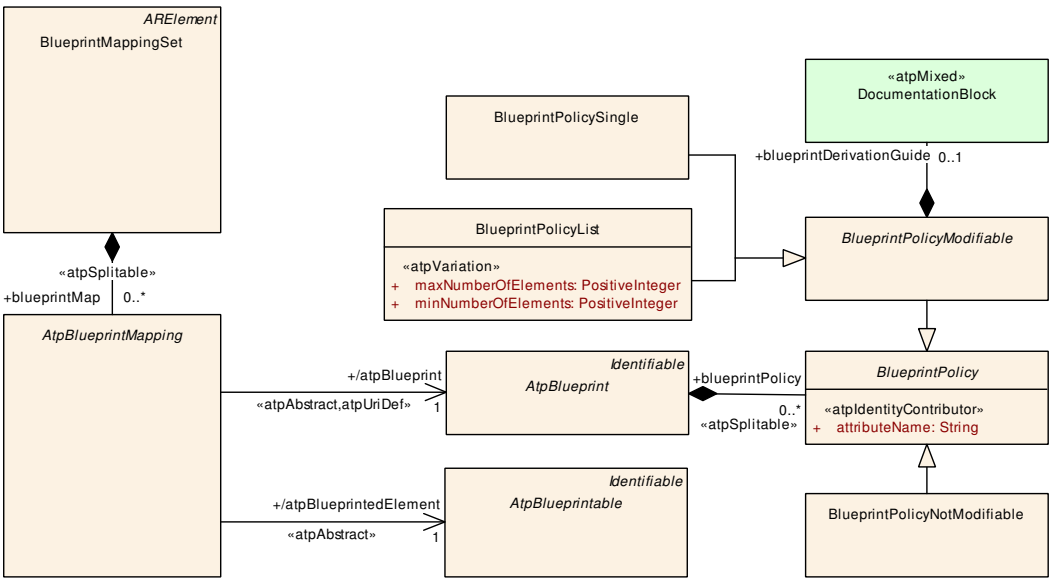


Figure 5.2: Abstract Blueprint Structure

Meta-classes for elements eligible for blueprinting are defined as specializations of `AtpBlueprintable` while meta-classes for blueprints are defined as specializations of `AtpBlueprint`. An example is given in figure 5.3.

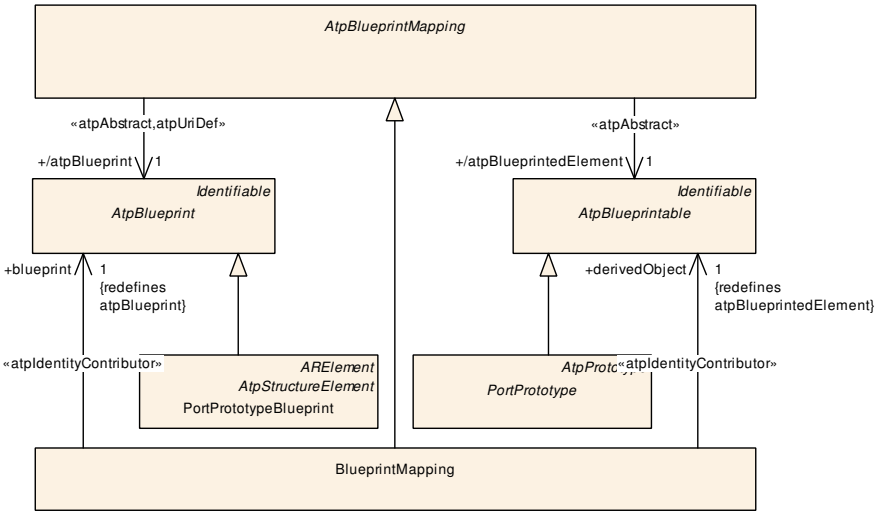


Figure 5.3: Port Blueprints as an example for separate meta-classes for Blueprint and blueprinted Element

[TPS_STDT_00072] Same Meta Class For Blueprints and Derived Objects ⌈For most of the elements eligible for blueprinting, no extra meta-class is required because the same meta-class applies for blueprints and blueprinted elements. The meta-class of such an element inherits from both `AtpBlueprint` and `AtpBlueprintable`.⌋

An example is given in figure 5.4.

[TPS_STDT_00041] Constraints may be violated in Blueprints ⌈For blueprints using the same meta-class as the derived objects, the constraints defined for these objects may be violated by the blueprints such as:

- Required attributes may be missing.
- Referenced objects may not exist. Strictly speaking, references in blueprints can all be considered as ≪`atpUriDef`≫

⌋

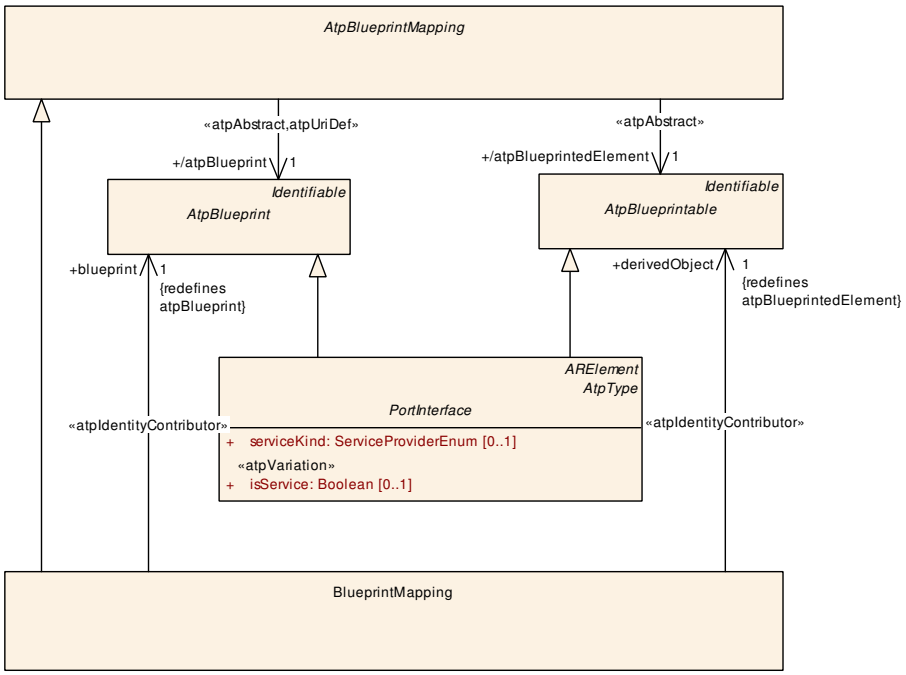


Figure 5.4: PortInterface Blueprints as an example for using the same meta-class for Blueprint and blueprinted Element

[TPS_STDT_00033] Recognize Blueprints ⌈According to [2] the blueprints reside in a package of `category` "BLUEPRINT". Downstream AUTOSAR Tools such as RTE-generator shall ignore Elements living in a package of `category` "BLUEPRINT".⌋

Blueprints are specializations of `AtpBlueprint`. Introduction of standardization therefore does not introduce compatibility problems to existing templates. Note that since AUTOSAR 4.0.3 `AtpBlueprint.shortNamePattern` is replaced by `Identifier.namePattern` resp. `CIdentifier.namePattern`. In addition since AUTOSAR 4.4.0 `blueprintValue` exists and is used e.g. in the context of ARMQL (AUTOSAR Model Query Language).

[TPS_STDT_00032] BlueprintPolicy ⌈Blueprintable elements shall be characterized by BlueprintPolicy to indicate whether they will be modifiable or not.

- `BlueprintPolicyNotModifiable` means, that the related attribute is not modifiable during the blueprinting
- `BlueprintPolicyList` means, that the related attribute is modifiable during the blueprinting. It applies only to an attribute with upper multiplicity > 1
- `BlueprintPolicySingle` means, that the related attribute is modifiable during the blueprinting. It applies only to an attribute with upper multiplicity == 1

⌋

Example ARXML listings for [TPS_STDT_00032] are shown in:

- `BlueprintPolicyNotModifiable`: A.8

- BlueprintPolicyList: A.9
- BlueprintPolicySingle: A.10

[constr_2590] One BlueprintPolicy is allowed ⌈For each attribute of a blueprint, at most one BlueprintPolicy is allowed.⌋

[constr_2591] BlueprintPolicyNotModifiable ⌈If BlueprintPolicyNotModifiable is assigned to an attribute, then during blueprinting it is not allowed to modify the value of the attribute and all its contained content.⌋

[constr_2592] No BlueprintPolicy ⌈If no BlueprintPolicy is assigned to an attribute, then arbitrary modifications are allowed while deriving from the blueprint.⌋

[constr_2593] Expression for identifying the attribute a BlueprintPolicy relates to ⌈The expression language for identifying the related attribute of a BlueprintPolicy is a subset version of xpath, see [13]. For navigation over the model we use the names as they are used in XML.⌋

[TPS_STDT_00039] Permitted XPath Expressions for BlueprintPolicy ⌈

XPath Expression	Description	Notes
nodename	Selects all nodes with the name "nodename"	
/	Selects from the root node (the root node is the blueprint owning the BlueprintPolicy)	
@	Selects attributes	
@<attribute>='<value>'	Selects an element node, which has the <attribute> set to <value>	
text()='<value>'	Selects an element node, which contains the text <value>	
*	Matches any element node	
[n]	Selects the n-th element node	Only allowed for ordered elements

⌋

The XPath expression [n] in [TPS_STDT_00039] starts with [1] due to [13]. One BlueprintPolicy can refine more than one attribute.

In listing A.8 the root node is selected by the nodename (COMPU-INTERNAL-TO-PHYS). In listing A.9 the root node is selected by nodename/nodename/* (COMPU-INTERNAL-TO-PHYS/COMPU-SCALES/*).

5.1.2 Mapping of Blueprints to blueprinted Elements

In many cases it will be necessary to identify the relationship of a blueprinted element (e.g. PortPrototype) to the corresponding blueprint (e.g. PortPrototypeBlueprint) after the blueprinted element has been created according to the blueprint.

For this purpose it would theoretically be possible to establish a reference from AtpBlueprintable to AtpBlueprint that identifies the pair of related model artifacts. However, this kind of information is relevant only in a narrow scope and does - as mentioned before - not impact the downstream model handling.

Therefore, a AtpBlueprintMapping is introduced which refers to both AtpBlueprintable and AtpBlueprint (see figure 5.2). The AtpBlueprintMapping is in turn aggregated at a container for the creation of blueprint mappings, the BlueprintMappingSet.

In previous AUTOSAR Releases a specialization of AtpBlueprintMapping was created for each particular meta class eligible for blueprinting. This has been replaced by one particular specialization (BlueprintMapping)[1].

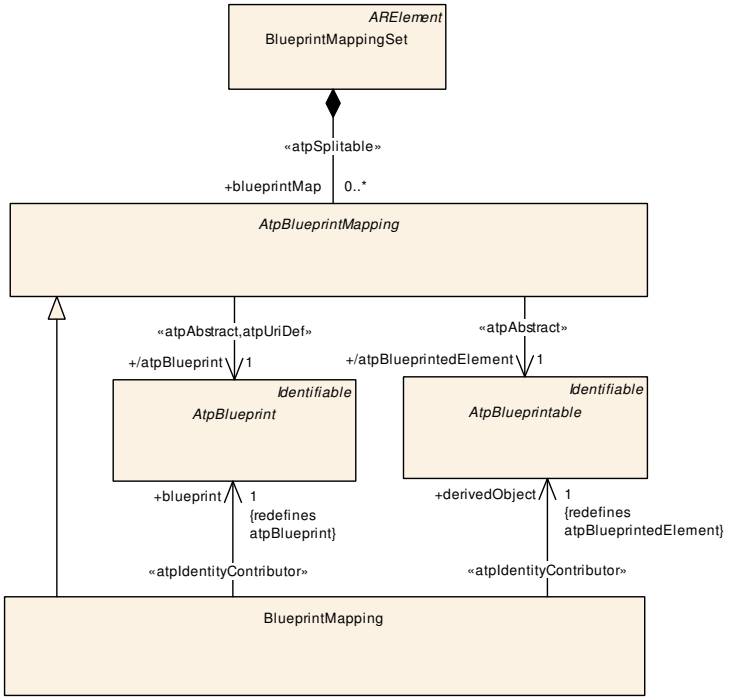


Figure 5.5: Mapping of Derived Objects and their Blueprints

[constr_2566] Blueprintmapping shall map appropriate elements ⌈ BlueprintMapping shall map elements which represent a valid pair of blueprint / derived object. In most of the cases this means that blueprint and derivedObject shall refer to objects of the same meta-class.⌋

Class	**BlueprintMappingSet**
Note	This represents a container of mappings between "actual" model elements and the "blueprint" that has been taken for their creation. **Tags:** atp.recommendedPackage=BlueprintMappingSets

▽

[1]For compatibility reasons, the abstract pattern was not changed. The previous specializations PortInterfaceBlueprintMapping and PortPrototypeBlueprintMapping are removed.

△

Class	**BlueprintMappingSet**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprintMap	AtpBlueprintMapping	*	aggr	This represents a particular blueprint map in the set. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=blueprintMap.blueprint, blueprint Map.derivedObject, blueprintMap.portInterfaceBlueprint, blueprintMap.portPrototypeBlueprint, blueprint Map.derivedPortInterface, blueprintMap.derivedPort Prototype

Table 5.1: BlueprintMappingSet

5.1.3 General Rules for Compliance of blueprint and blueprinted element

[TPS_STDT_00005] Compliance with Blueprints ⌈Constraints [constr_2554] and [TPS_STDT_00087] apply in general for the compliance of blueprints with the derived objects.⌋

[constr_2554] Derived objects shall match the blueprints ⌈Unless specified explicitly otherwise, the attributes of the blueprint shall appear in the derived objects. As an exception `namePattern` and `blueprintValue` may **not** be copied.⌋

[TPS_STDT_00087] Derived objects may have more attributes than the blueprints ⌈Unless specified explicitly otherwise, derived objects may have more attributes than the blueprints. Such attributes can be

- additional values if the upper multiplicity of the attribute in the meta-model is greater than 1
- those specified by the related templates but not specified in the blueprint

⌋

[TPS_STDT_00085] Compatibility of `longName`, `desc` and `introduction` of blueprint and blueprinted element ⌈Elements derived from blueprints are allowed to:

- change `longName`
- change `desc`
- change `introduction`

⌋

Note that [TPS_STDT_00085] includes the ability to add text in a further language.

Note that introduction should not be used to describe the derivation of objects from the blueprint. See [TPS_STDT_00048] for details.

[TPS_STDT_00086] Specify a name pattern or a blueprint value in blueprints ⌈For each blueprint, a namePattern or a blueprintValue shall be specified if the shortName respectively a symbol is not fixed but intended to be defined when objects are derived from a blueprint. This is used to verify the appropriate naming of the derived objects ([constr_2553]).⌋

[constr_2553] shortName shall follow the pattern defined in the Blueprint ⌈The shortName respectively symbol of the derived objects shall follow the pattern defined in namePattern or blueprintValue of the blueprint according to [TPS_STDT_00086]⌋

[constr_2570] No Blueprints in system descriptions ⌈There shall be no blueprints in system descriptions. In consequence of this blueprint elements shall be referenced only from blueprints and AtpBlueprintMappings. Due to ≪atpUriDef≫, the references from AtpBlueprintMapping do not need to be resolved in system descriptions.⌋

[constr_2571] Outgoing references from Blueprints ⌈Note that outgoing references from Blueprints are basically not limited. Practically, references to objects living in a package of category EXAMPLE should not occur.⌋

Reason for [constr_2571] is the fact that these examples then also shall exist in the target system description but not as example. In such a case the example would take the role of a blueprint.

Figure 5.6 illustrates a scenario with standardized objects, blueprints and project related objects.

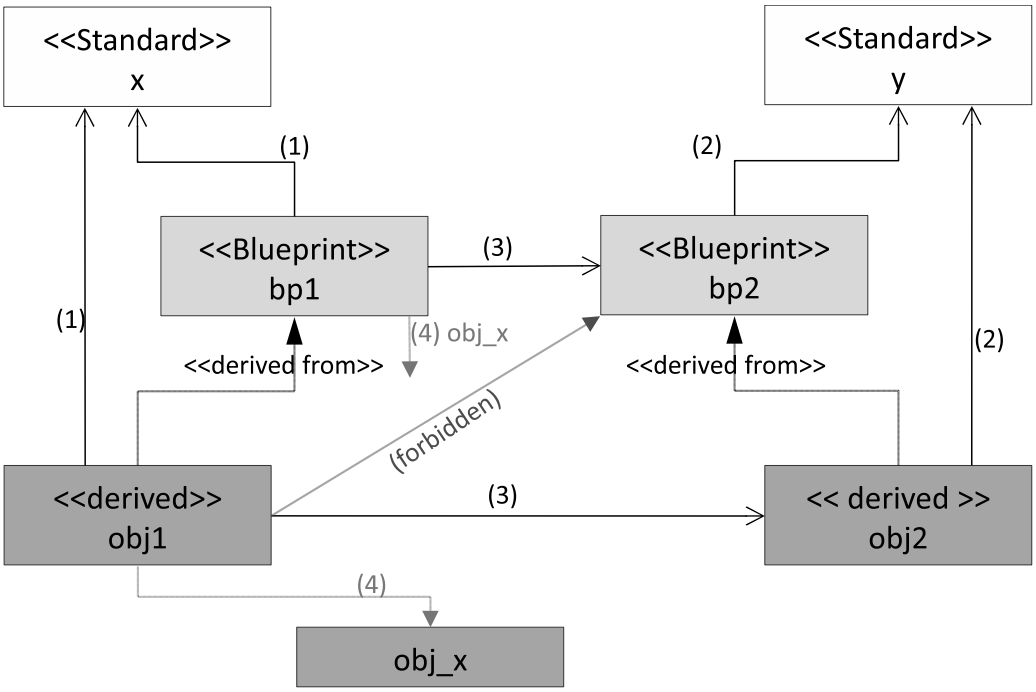


Figure 5.6: Relations between Blueprints, "Derived Objects" and "Standardized Objects"

This diagram in particular illustrates how references in blueprints shall be handled:

[TPS_STDT_00051] Handling references when deriving objects from blueprints ⌈

- Blueprints may reference standardized objects. These references also exist in the derived objects (1), (2).
- Blueprints may reference other blueprints (3). These references need to be replaced in order to meet [constr_2546]. Therefore a reference from a derived object to a blueprint is not allowed.
- Blueprints may contain references to arbitrary objects (4). According to [TPS_STDT_00041] it is allowed that these objects even do not exist. Nevertheless to meet [constr_2554] such references shall be copied to the derived objects and the referenced objects shall exist in the target system description.

⌋

[TPS_STDT_00034] Integrity of Blueprints ⌈The integrity of blueprints can be established by applying references to blueprints of related objects. For example, a blueprint of a `BswModuleDescription` may refer to a blueprint of `BswModuleEntry`.⌋

[constr_2546] References in derived model elements ⌈Model elements derived from blueprints shall never refer to model elements that are blueprints.⌋

Note: A blueprint may refer to another blueprint. When deriving objects such a reference shall be replaced such that the new reference target is an object derived from the corresponding reference target in the blueprint.

[TPS_STDT_00065] Nested Blueprint Can be Used as Blueprint of its own ⌈If specialization of AtpBlueprint aggregates specialization of AtpBlueprint, then the such aggregated specialization of AtpBlueprint acts as a blueprint on its own and can be derived beyond the context of objects derived from the aggregating specialization of AtpBlueprint. This definition allows to create blueprints which are not specializations of ARElement.

In other words, If a blueprint contains blueprints, the "inner" blueprints can be derived independent from derived objects of the "outer" blueprint.⌋

See chapter 5.2.8 for an use case of [TPS_STDT_00065].

[TPS_STDT_00047] Ignore Blueprint Attributes in Non Blueprints ⌈AUTOSAR Tools which do not process blueprints such as RTE-generator shall ignore Identifier.namePattern resp. CIdentifier.namePattern and blueprintValue.

The attributes Identifier.namePattern resp. CIdentifier.namePattern and blueprintValue should be removed when deriving objects from blueprints.⌋

[TPS_STDT_00048] Express Decisions when Deriving Objects ⌈Applying VariationPoint is a suitable way to express intended decisions to be made when deriving objects from blueprints. In this case the value of the UML tag vh.latestBindingTime is blueprintDerivationTime and VariationPoint.blueprintCondition, VariationPoint.formalBlueprintGenerator respectively AttributeValueVariationPoint.blueprintValue shall be used to express the intended derivation.⌋

[TPS_STDT_00028] Resolving VariationPoint in Blueprints ⌈If a VariationPoint has only blueprintValue respectively blueprintCondition, formalBlueprintGenerator but not swSyscond nor postBuildVariantCondition it shall be resolved when deriving elements.⌋

Please refer to Generic Structure Template [2] for the following aspects:

- Even if BindingTimeEnum does not contain the value blueprintDerivationTime, there are still VariationPoints which shall be bound on blueprint derivation. This is specified as blueprintDerivationTime in the UML tag vh.latestBindingTime at the variation point in the meta-model.

  See chapter 5.2 for such elements.

- See [constr_2557]: System configurations shall not contain VariationPoints with vh.latestBindingTime set to blueprintDerivationTime.

- [constr_2558]: If vh.latestBindingTime is blueprintDerivationTime then there shall only be blueprintCondition, formalBlueprintGenerator respectively blueprintValue.

- See [constr_2559]: `VariationPoint`s shall not be nested. In particular this means that there shall not exist a `VariationPoint` within the `DocumentationBlock` in the role `blueprintCondition` in a `VariationPoint`.
- See [constr_2567]: Attribute Value Blueprints should contain `undefined`.

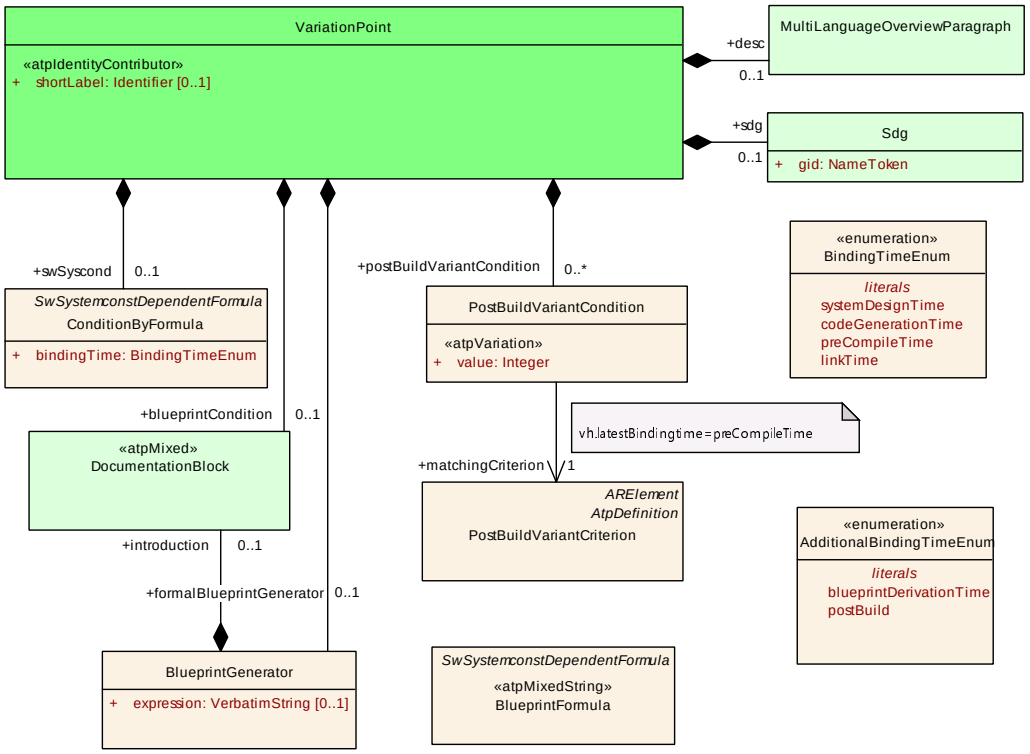


Figure 5.7: Variation Point

Class	**VariationPoint**			
Note	This meta-class represents the ability to express a "structural variation point". The container of the variation point is part of the selected variant if swSyscond evaluates to true and each postBuildVariant Criterion is fulfilled.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprint Condition	DocumentationBlock	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint. Note that variationPoints are not allowed within a blueprintCondition. **Tags:** xml.sequenceOffset=28
desc	MultiLanguageOverview Paragraph	0..1	aggr	This allows to describe shortly the purpose of the variation point. **Tags:** xml.sequenceOffset=20

▽

△

Class	VariationPoint			
formalBlueprint Generator	BlueprintGenerator	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint by using ARMQL. Note that variationPoints are not allowed within a formal BlueprintGenerator. **Tags:** atp.Status=draft xml.sequenceOffset=30
postBuildVariant Condition	PostBuildVariant Condition	*	aggr	This is the set of post build variant conditions which all shall be fulfilled in order to (postbuild) bind the variation point. **Tags:** xml.sequenceOffset=40
sdg	Sdg	0..1	aggr	An optional special data group is attached to every variation point. These data can be used by external software systems to attach application specific data. For example, a variant management system might add an identifier, an URL or a specific classifier. **Tags:** xml.sequenceOffset=50
shortLabel	Identifier	0..1	attr	This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splitable aggregations and if binding time is later than codeGenerationTime, as well as some RTE conditions. It needs to be unique with in the enclosing Identifiables with the same ShortName. **Stereotypes:** atpIdentityContributor **Tags:** xml.sequenceOffset=10
swSyscond	ConditionByFormula	0..1	aggr	This condition acts as Binding Function for the Variation Point. Note that the multiplicity is 0..1 in order to support pure postBuild variants. **Tags:** xml.sequenceOffset=30

Table 5.2: VariationPoint

[TPS_STDT_00030] Blueprint of `VariationPoint` ⌈A blueprint may contain `VariationPoint` with `vh.latestBindingTime` set to `blueprintDerivationTime`. These are considered as kind of blueprint of variation points which shall be handled when deriving objects. The following options apply for the container of the `VariationPoint` according to chosen approach for blueprint derivation:

1. If `blueprintCondition` is specified: resolved manually
2. If `formalBlueprintGenerator` is specified: resolved by a module generator. The resolver approach is formalized using ARMQL. Note that in this case it is also likely that multiple objects are created by the module generator.

After resolving the `VariationPoint` by one of these conditions the remaining variation is converted to a subsequent `VariationPoint`.⌋

[TPS_STDT_00044] Transferring `VariationPoint` ⌈Unless specified explicitly otherwise, `VariationPoint`s with `vh.latestBindingTime` **not** set to `BlueprintDerivationTime` should be transferred to the derived objects (see also [TPS_STDT_00087]). Thereby the `shortLabel` of the `VariationPoint` may be adapted according to the specification in the `blueprintCondition` and `formalBlueprintGenerator`.⌋

[constr_2556] No Blueprint Motivated `VariationPoint`s in AUTOSAR Descriptions ⌈AUTOSAR descriptions which are not blueprints shall not have `blueprintCondition`, `formalBlueprintGenerator` nor `blueprintValue`.⌋

[constr_2569] Purely Blueprint Motivated `VariationPoint`s ⌈`VariationPoint`s with `vh.latestBindingTime` set to `blueprintDerivationTime` shall have only `blueprintCondition` or `formalBlueprintGenerator` respectively `blueprintValue`.⌋

[TPS_STDT_00045] Transferring Objects in General ⌈Objects resp. references without `VariationPoint` shall be transferred to the derived objects. Thereby the `namePattern`s and the `blueprintValue`s of the referenced Blueprints also apply for rewriting the shortName path in the reference.⌋

For more details about `VariationPoint` refer to [2], as all constraints are summarized there.

[TPS_STDT_00046] Configuration dependent properties ⌈Some data types specify configuration-dependent properties like limits, base types etc. This is supported by an additional attribute `blueprintValue` in the `AttributeValueVariationPoint`.⌋

An example for [TPS_STDT_00046] is:

```
NvM_BlockIdType Range:  0..2\^(16- NvMDatasetSelectionBits)-1
Dem_RatioIdType Type: uint8, uint16
```

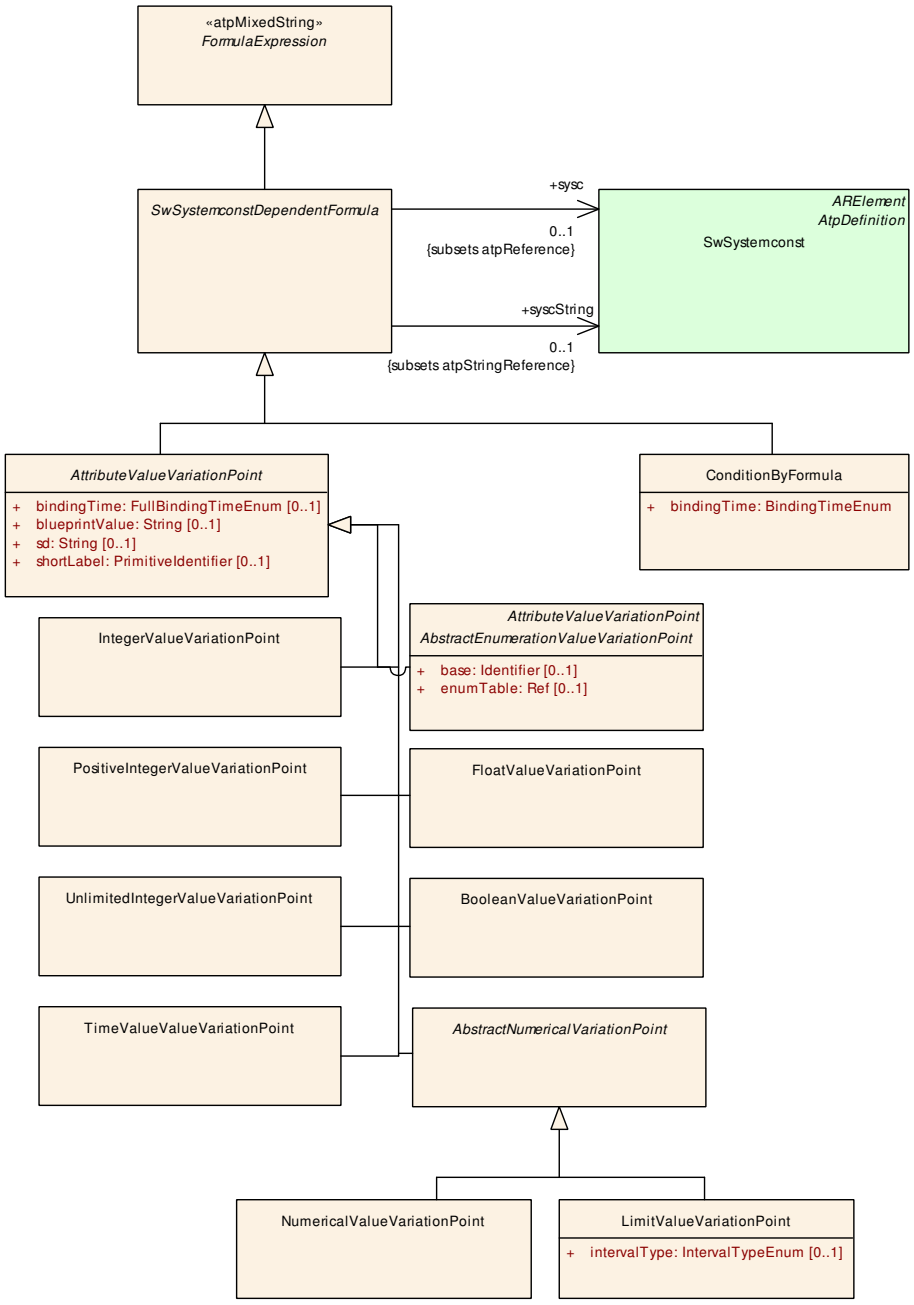


Figure 5.8: Attribute Value Variation Point

Class	«atpMixedString» ***AttributeValueVariationPoint*** (abstract)			
Note	This class represents the ability to derive the value of the Attribute from a system constant (by Sw SystemconstDependentFormula). It also provides a bindingTime.			
Base	*ARObject*, *FormulaExpression*, *SwSystemconstDependentFormula*			
Subclasses	*AbstractEnumerationValueVariationPoint*, *AbstractNumericalVariationPoint*, BooleanValueVariationPoint, FloatValueVariationPoint, IntegerValueVariationPoint, PositiveIntegerValueVariationPoint, TimeValue ValueVariationPoint, UnlimitedIntegerValueVariationPoint			
Aggregated by	VariationPointProxy.valueAccess			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	«atpMixedString» ***AttributeValueVariationPoint*** (abstract)			
bindingTime	FullBindingTimeEnum	0..1	attr	This is the binding time in which the attribute value needs to be bound. If this attribute is missing, the attribute is not a variation point. In particular this means that It needs to be a single value according to the type specified in the pure model. It is an error if it is still a formula. **Tags:** xml.attribute=true
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:** xml.attribute=true
sd	String	0..1	attr	This special data is provided to allow synchronization of Attribute value variation points with variant management systems. The usage is subject of agreement between the involved parties. **Tags:** xml.attribute=true
shortLabel	PrimitiveIdentifier	0..1	attr	This allows to identify the variation point. It is also intended to allow RTE support for CompileTime Variation points. **Tags:** xml.attribute=true

Table 5.3: AttributeValueVariationPoint

5.1.4 Applicable patterns to define attributes when deriving objects from blueprints

5.1.5 Name Patterns

[TPS_STDT_00003] Applying `namePattern` ⌈When deriving an element from a blueprint it is often the case that a particular pattern shall be used to determine the `shortName` respectively the `symbol` of the object. This use case is supported by the attribute `namePattern` in `Identifier` respectively `CIdentifier`.⌋

Primitive	**Identifier**			
Note	An Identifier is a string with a number of constraints on its appearance, satisfying the requirements typical programming languages define for their Identifiers. This datatype represents a string, that can be used as a c-Identifier. It shall start with a letter, may consist of letters, digits and underscores. **Tags:** xml.xsd.customType=IDENTIFIER xml.xsd.maxLength=128 xml.xsd.pattern=[a-zA-Z][a-zA-Z0-9_]* xml.xsd.type=string			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:** atp.Status=draft xml.attribute=true

▽

△

Primitive	**Identifier**			
namePattern	String	0..1	attr	This attribute represents a pattern which shall be used to define the value of the identifier if the identifier in question is part of a blueprint. For more details refer to TPS_StandardizationTemplate. **Tags:** xml.attribute=true

Table 5.4: Identifier

Primitive	**CIdentifier**			
Note	This datatype represents a string, that follows the rules of C-identifiers. **Tags:** xml.xsd.customType=C-IDENTIFIER xml.xsd.pattern=[a-zA-Z_][a-zA-Z0-9_]* xml.xsd.type=string			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
blueprintValue	String	1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:** atp.Status=draft xml.attribute=true
namePattern	String	0..1	attr	This attribute represents a pattern which shall be used to define the value of the identifier if the CIdentifier in question is part of a blueprint. For more details refer to TPS_StandardizationTemplate. **Tags:** xml.attribute=true

Table 5.5: CIdentifier

[TPS_STDT_00055] General Syntax for Name Patterns ⌈The name pattern uses the syntax described in ANTLR [14].⌋

An example ARXML listings for [TPS_STDT_00055] is:

```
grammar NamePattern;

options { language = Ruby;
          output = AST;}

namePattern
        : (fixedName | placeholder | separator)+ ;

subPattern
  : '('  (fixedName | placeholder | separator )+  ')' ('?' | '*' | '+')? ;


placeholder  : '{'
                 ('anyName' |
                  'anyNamePart' |
                  'blueprintName' |
                  'capitalizedCallbackName' |
                  'capitalizedMip' |
                  'codePeriode' |
                  'componentName' |
                  'componentTypeName' |
                  'componentPrototypeName' |
```

```
                'ecucValue' '(' ecucName ')' |
                'index' |
                'initPolicy' |
                'keyword' '(' kwClass ')' |
                'Mip' |
                'modeName' |
                'nameSpace' |
                'portDir' |
                'typeId'    |
                subPattern
              )
            '}' ;

fixedName :  MyName;

kwClass :    MyName;

separator
  :    Separator ;

pathSeparator
       : PathSeparator ;

ecucName:     ( anyNamePart | pathSeparator)+;

anyNamePart :    MyName (separator MyName)*;

MyName     :    ( 'a'..'z' | ('A'..'Z') | ('0'..'9') | '-' )*;


Separator :  '_' ;

PathSeparator : '/' ;
```

Listing 5.1: Grammar for name pattern

This example illustrates valid name patterns. Note that `{blueprintName}` etc. denotes a placeholder.

```
{blueprintName}_{anyName}

{portDir}_{blueprintName}_{keyword(Qualifier)}_{componentName}_{index}
  --> example for a match:  R_EngN_Max_Dem_3

{componentName}_{ecucValue(item1)}

h_b_{(a_{index}_b_{componentName}_{(x_{ecucValue(hugo)})*})*}
```

The semantics of the placeholder is defined as follows:

anyName This represents a string which is valid `shortName` according to `Identifier`

anyNamePart This represents a string [a-zA-Z0-9_]* which is valid part of a shortName.

Hint: The place holder "anyNamePart" shall not be used at the beginning of a shortName pattern to avoid invalid shortNames.

blueprintName This represents the shortName / shortLabel / symbol of the applied blueprint

capitalizedCallbackName This represents the name of the callback function including module prefix, but written in upper case.

capitalizedMip This represents the capitalized module implementation prefix according to [SWS_BSW_00102]. All characters are converted to uppercase.

codePeriode This represents the period time value and unit. Units are: US micro seconds, MS milliseconds, S second. For example: 100US, 10MS, 1S.

componentName This represents the shortName of the BSW module resp. ASW SwComponentType / ASW component prototype related to the derived object. "Related" mainly could be both, aggregating or referencing.

[TPS_STDT_00036] Placeholder for Module / Component ⌈The placeholder componentName in particular supports multiple derivation of a PortPrototypeBlueprint in the context of different software component types resp. modules.⌋

componentTypeName This represents the shortName of the dedicated SwComponentType.

componentPrototypeName This represents the shortName of the dedicated SwComponentPrototype.

ecucValue [TPS_STDT_00040] Influence of ECUC ⌈This indicates an influence of the ECU configuration. This placeholder takes an argument which is intended as a keyword reflecting the kind of influence. More details shall be specified in the blueprintCondition where the argument mentioned before can be taken for reference.⌋

index This represents a numerical index applicable for example to arrays.

initPolicy This represents the initialization policy of variables according to SectionInitializationPolicyType where the dashes are replaced by underscores, e.g. NO_INIT, CLEARED, POWER_ON_CLEARED, INIT, POWER_ON_INIT.

keyword [TPS_STDT_00004] Abbreviated Name ⌈This represents the abbrName of a keyword acting as a name part of the short name. The eligible keywords can be classified (using the argument kwClass). This classification shall match with one of the classification of the applied keyword.⌋

Mip This represents the module implementation prefix according to [SWS_BSW_-00102].

modeName This represents the shortName of the mode e.g. `Dcm_{modeName}ModeEntry`

portDir This represents the direction of a port.

> **[TPS_STDT_00037] Port Direction** ⌈The placeholder `portDir` in particular supports the case that the same blueprint is used for P-Port as well as for an R-Port. The values represented by this placeholder is `P` for P-Port respectively `R` for R-Port.⌋

typeId This represents an indicator based on the type of the object.

5.1.6 Blueprint Formula

[TPS_STDT_00006] Applying Expression Pattern ⌈When deriving an element from a blueprint it is often the case that a particular pattern shall be used to determine the value and or the condition of the object. This use case is supported by the attribute `blueprintValue`.⌋

[TPS_STDT_00010] General Syntax for Expression Patterns ⌈The expression pattern uses the syntax of the Formula Language as defined in [TPS_GST_00012].⌋

[TPS_STDT_00092] Return values of the `BlueprintFormula.ecuc` query ⌈

Return values	Description
`EcucContainerDef`	Ecuc returns the value of the shortName of the `EcucContainerValue`
`EcucBooleanParamDef`	Ecuc returns the assigned value of the `EcucNumericalParamValue`
`EcucIntegerParamDef`	Ecuc returns the assigned value of the `EcucNumericalParamValue`
`EcucFloatParamDef`	Ecuc returns the assigned value of the `EcucNumericalParamValue`
`EcucEnumerationParamDef`	Ecuc returns the assigned value of the `EcucTextualParamValue`
`EcucAbstractString-ParamDef`	Ecuc returns the assigned value of the `EcucTextualParamValue`
`EcucReferenceDef`	Ecuc returns the referenced container object qualified by the destination attribute
`EcucChoiceReferenceDef`	Ecuc returns the referenced container objects (list) qualified by the destination attributes
`EcucUriReferenceDef`	Ecuc returns the referenced container objects (list) qualified by the destinationUri attribute

If several `EcucContainerValue`(s) or `EcucParameterValue`(s) are assigned to the `EcucContainerDef` / `EcucParameterDef` the return value is undefined.

⌋

[TPS_STDT_00021] Specialization of `BlueprintFormula` ⌈These specialization(s) express the extension of the Formula Language to provide formalized `blueprint-Value`:

- ecuc: queries to the values described for ECUC-DEFINITION-ELEMENT. Depending on the ECUC-DEFINITION-ELEMENT a value or a string or an object is the result, see [TPS_STDT_00092]
- sysc: queries to the values assigned to SW-SYSTEMCONST
- syscString: indicates that the referenced system constant shall be evaluated as a string according to [TPS_SWCT_01431]
- <VERBATIM>: defines the ability to specify non formula parts
- ->: Reference Operator; a -> b the value of object 'b' as specified in [TPS_STDT_00092] which is pointed to by 'a'

⌋

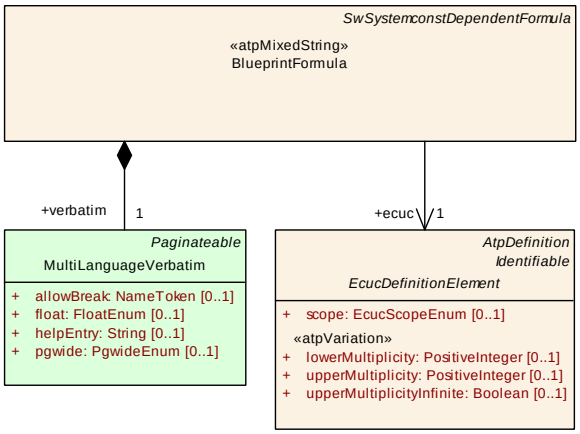


Figure 5.9: Blueprint Formula

Listing 5.2 illustrates valid expression patterns. Note that `blueprintValue` denotes a placeholder.

```
<EXPRESSION xml:space="preserve">
FOR
  configClass    : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");
LET
  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";
  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";
WHERE
  isConfigClass2 OR isConfigClass3;
```

Listing 5.2: Use of Logical Expression

In listing 5.3 the use of the Reference Operator is illustrated. The Reference Operator is inserted as a XML entity.

```
<!-- example for Reference Operator -->
<VARIATION-POINT>

  <!-- <FORMAL-BLUEPRINT-CONDITION>
```

Listing 5.3: Use of Reference Operator

5.1.7 Ecu Configuration Parameters and Blueprints

[TPS_STDT_00025] Deriving VSMD from STMD Uses its own Mechanism ⌈Basically the Standard Module Definitions (STMD) specified by AUTOSAR according to [15] could also be considered as blueprints. On the other hand, the relationship between vendor specific module definitions (VSMD) is a very strict one and was there before the general concept of Blueprints was introduced. Therefore for sake of compatibility this relationship is still maintained using `refinedModuleDef`.

Nevertheless for company specific applications there is some support for ECU configuration in Standardization Template.⌋

See chapter 5.2.13 resp. chapter 5.2.14 for more details.

5.2 Blueprintables defined in AUTOSAR Meta Model

The following sub chapters specify the particular model elements for which blueprints are supported.

5.2.1 Blueprinting AccessControl

[TPS_STDT_00062] Blueprinting Elements of AccessControl ⌈`AclObjectSet`, `AclOperation`, `AclPermission`, `AclRole` can be blueprinted.⌋

5.2.2 Blueprinting AliasNameSet

[TPS_STDT_00011] Blueprinting `AliasNameSet` ⌈`AliasNameSet` can be blueprinted.⌋

5.2.3 Blueprinting ApplicationDataType

[TPS_STDT_00023] Blueprinting `ApplicationDataType` ⌈`ApplicationDataType` can be blueprinted.⌋

5.2.4 Blueprinting ARPackage

[TPS_STDT_00013] Blueprinting `ARPackage` ⌈`ARPackage` can be blueprinted. Main use case is to support predefined package structures, e.g. those specified in [2].⌋

5.2.5 Blueprinting BswModuleDescription

[TPS_STDT_00027] Blueprinting `BswModuleDescription` ⌈`BswModuleDescription` can be blueprinted.⌋

Blueprints for `BswModuleDescription` are used in particular to describe dependencies to other modules. Note that in this case all references to other modules and module entries are targeting blueprints of the intended module. These references need to be replaced when deriving objects from the blueprint of `BswModuleDescription`.

A blueprint of `BswModuleDescription` shall specify the references to the standard- or blueprint- API elements, in particular

- `BswModuleDescription.implementedEntry`
- `BswModuleDescription.expectedEntry`

Nevertheless, it is allowed that derived `BswModuleDescription` adds further ones of these references.

Furthermore, optional elements like callbacks often come in 0..* multiplicity. In this case, the blueprint should specify one callback reference (to one blueprint BswModuleEntry) and express the open multiplicity in its `namePattern` respectively in the `VariationPoint.blueprintCondition` or `VariationPoint.formalBlueprintGenerator` as illustrated in Figure 5.10.

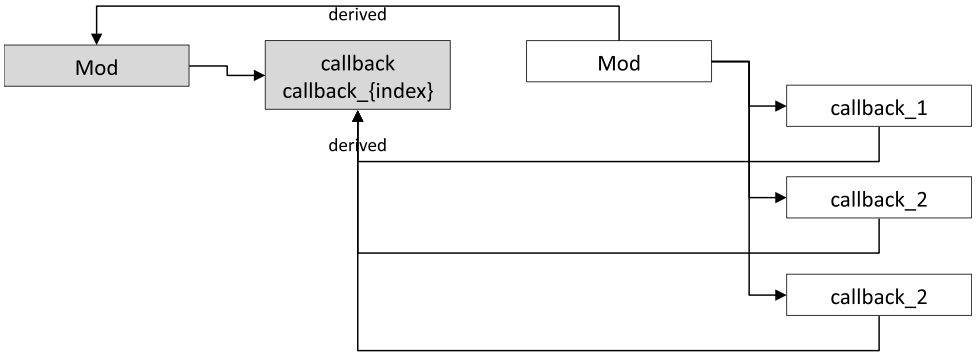


Figure 5.10: Multiply derived Objects

[constr_2563] `BswModuleDescription` blueprints should not have a `BswInternalBehavior` ⌈A `BswModuleDescription` blueprint should not have a `BswInternalBehavior` since this is a matter of implementation and not subject to standardization. Exceptions might exist in vendor internal applications.⌋

5.2.6 Blueprinting BswModuleEntry

[TPS_STDT_00014] Blueprinting `BswModuleEntry` ⌈`BswModuleEntry` can be blueprinted.⌋

The meta-class BswModuleEntry and its composites (SwServiceArg) contain optional as well as mandatory elements which are never or only sometimes standardized, e.g. executionContext, swServiceImplPolicy, parts of SwServiceArg.swDataDefProps. Nevertheless Standardization Template does not explicitly specify constraint which attributes shall, may or shall not be defined in the blueprint (see also [TPS_STDT_00049]).

5.2.7 Blueprinting BswEntryRelationshipSet

[TPS_STDT_00090] Blueprinting BswEntryRelationshipSet ⌈BswEntryRelationshipSet can be blueprinted.⌋

[TPS_STDT_00091] Blueprinting BswEntryRelationshipSet ⌈The BswEntryRelationshipSet describes a collection of BswEntryRelationships. A BswEntryRelationship describes a relationship between two BswModuleEntrys and the type of relationship. This is typically used to express that a concrete BswModuleEntry is derived from an abstract BswModuleEntry. In this case the bswEntryRelationshipType is set to derivedFrom, the BswEntryRelationship.from references the abstract BswModuleEntry and the BswEntryRelationship.to references the concrete BswModuleEntry.⌋

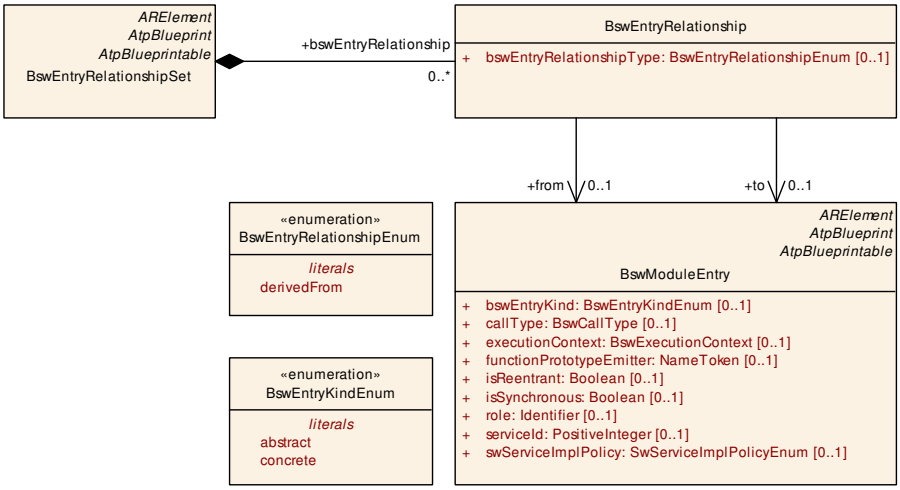


Figure 5.11: BswEntryRelationshipSet

Class	**BswEntryRelationshipSet**			
Note	Describes a set of relationships between two BswModuleEntrys. **Tags:** atp.recommendedPackage=BswEntryRelationshipSets This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswEntry Relationship	BswEntryRelationship	*	aggr	Relationship between two BswModuleEntrys.

Table 5.6: BswEntryRelationshipSet

Class	**BswEntryRelationship**			
Note	Describes a relationship between two BswModuleEntrys and the type of relationship.			
Base	*ARObject*			
Aggregated by	BswEntryRelationshipSet.bswEntryRelationship			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswEntry Relationship Type	BswEntryRelationship Enum	0..1	attr	Denotes the type of the relationship. **Tags:** xml.sequenceOffset=5
from	BswModuleEntry	0..1	ref	Type of relationship that refers to the abstract BswModule Entry. Please notice that in this case the bswEntry RelationshipType shall be set to drivedFrom. This Attribute is only used by the AUTOSAR Classic Platform.
to	BswModuleEntry	0..1	ref	Type of relationship that refers to the concrete Bsw ModuleEntry This Attribute is only used by the AUTOSAR Classic Platform.

Table 5.7: BswEntryRelationship

Enumeration	**BswEntryRelationshipEnum**
Note	Define the type of relationship between two BswModuleEntrys.
Aggregated by	BswEntryRelationship.bswEntryRelationshipType
Literal	***Description***
derivedFrom	Describes that the BswModuleEntry referenced as "to" needs to have the same signature as the "abstract" BswModuleEntry referenced as "from". **Tags:** atp.EnumerationLiteralIndex=0

Table 5.8: BswEntryRelationshipEnum

5.2.8 Blueprinting BuildActionManifest

[TPS_STDT_00063] Blueprinting `BuildActionManifest` ⌈`BuildActionManifest` can be blueprinted. [TPS_STDT_00065] applies such that blueprints of `BuildAction` and `BuildActionEnvironment`s are aggregated in a blueprint of `BuildActionManifest`.⌋

5.2.9 Blueprinting CompuMethod

[TPS_STDT_00015] Blueprinting `CompuMethod` ⌈`CompuMethod` can be blueprinted.⌋

Sometimes it is required to extend a standardized enumeration with vendor specific elements.

For example `[SWS_RamTst_00192]` states: If vendor specific algorithms were defined the enumeration fields of RamTst_AlgorithmType should be extended accordingly.

[TPS_STDT_00049] Blueprinting Enumerators ⌈Extensions of enumerator values shall be expressed in the blueprint of the related `CompuMethod` by the `VariationPoint` at `CompuScale`.⌋

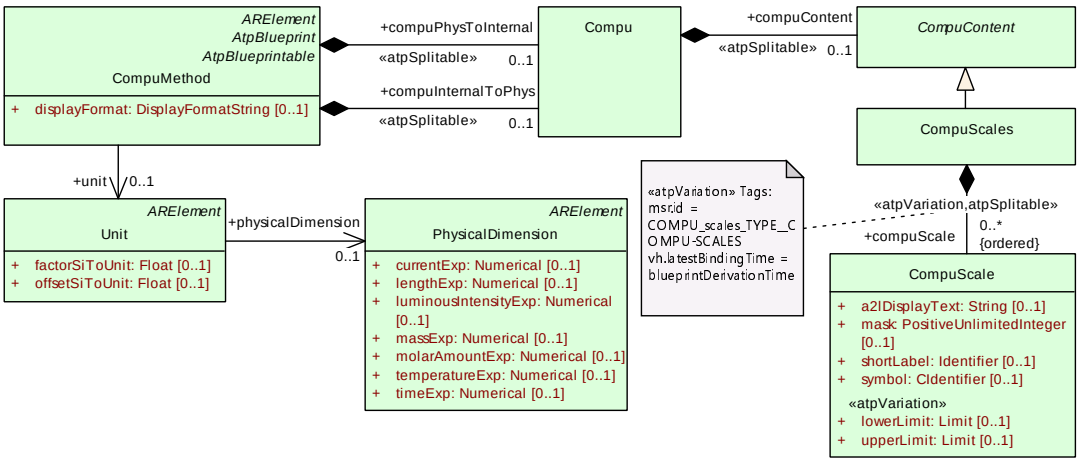


Figure 5.12: A `CompuMethod` and its attributes define data semantics

5.2.10 Blueprinting ConsistencyNeeds

[TPS_STDT_00071] Blueprinting `ConsistencyNeeds` ⌈`ConsistencyNeeds` can be blueprinted. But as it is not derived from `ARElement`, all such blueprints are aggregated by `ConsistencyNeedsBlueprintSet`. This allows to apply [TPS_STDT_00072].⌋

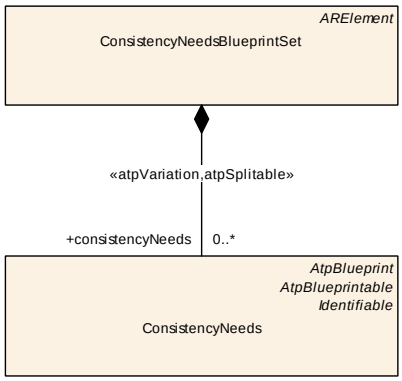


Figure 5.13: Blueprinting `ConsistencyNeeds`

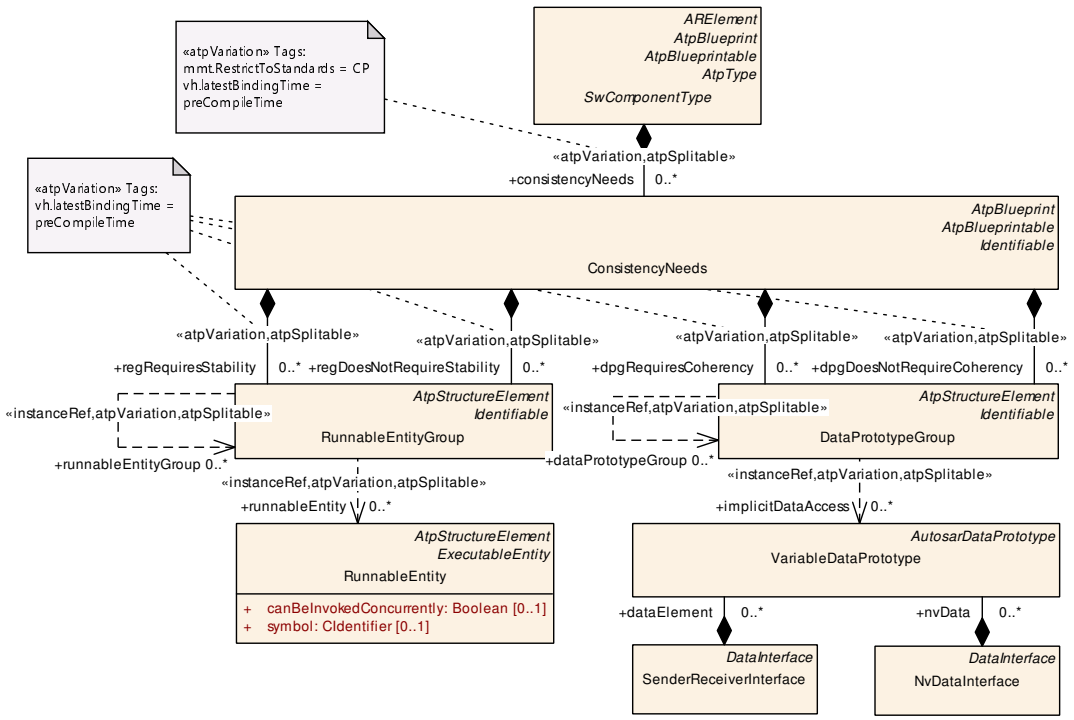


Figure 5.14: `ConsistencyNeeds`

[TPS_STDT_00073] Early definition of ConsistencyNeeds ⌈Grouping of Data shall be possible before the `RunnableEntity`s with all the details (data access points) are known. In a top down approach the grouping of DataPrototypes can already be used to design the system in a way that consistency properties are guaranteed and that consistency is not required for unrelated DataPrototypes.

Therefore the `DataPrototypeGroup` in a `ConsistencyNeeds` (Blueprint) can reference `VariableDataPrototype`s of `PortInterface`s without any further context information.⌋

[TPS_STDT_00074] Categorization of Blueprints of `ConsistencyNeeds` ⌈Since a `ConsistencyNeeds`(Blueprint) can be designed before the software component is known in all details it is required to denote the purpose of the `DataPrototypeGroup` and the `RunnableEntityGroup` of a `ConsistencyNeeds`(Blueprint). Therefore a

set of category values is predefined which supports the "abstract" blueprinting of ConsistencyNeeds.⌋

[TPS_STDT_00075] Categories for DataPrototypeGroup in a Blueprint of ConsistencyNeeds ⌈

ALL_PROVIDE_DATA_OF_COMPONENT DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes instantiated in provide ports of the software component.

ALL_REQUIRE_DATA_OF_COMPONENT DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes instantiated in require ports of the software component.

ALL_PROVIDE_AND_REQUIRE_DATA_OF_COMPONENT DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes instantiated in provide and require ports of the software component.

ALL_PROVIDE_DATA_OF_RUNNABLE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes where any RunnableEntity in the attached RunnableEntityGroup has a implicit write access to it.

ALL_REQUIRE_DATA_OF_RUNNABLE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes where any RunnableEntity in the attached RunnableEntityGroup has a implicit read access to it.

ALL_PROVIDE_AND_REQUIRE_PORTS_OF_RUNNABLE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes where any RunnableEntity in the attached RunnableEntityGroup has a implicit write or read access to it.

EXPLICIT_DATA_PROTOTYPE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain VariableDataPrototypes according functional requirements

⌋

[TPS_STDT_00076] Categories for RunnableEntityGroup in a Blueprint of ConsistencyNeeds ⌈

ALL_RUNNABLES_OF_COMPONENT RunnableEntityGroup of the ConsistencyNeeds shall contain all RunnableEntitys of the software component.

ALL_RUNNABLES_WRITING_TO_DATA_PROTOTYP_GROUP RunnableEntityGroup of the ConsistencyNeeds shall contain all RunnableEntitys with a implicit write access to any of the VariableDataPrototypes in the attached DataPrototypeGroup.

ALL_RUNNABLES_READING_FROM_DATA_PROTOTYPE_GROUP RunnableEntityGroup of the ConsistencyNeeds shall contain all RunnableEntitys with a implicit read access to any of the VariableDataPrototypes in the attached DataPrototypeGroup.

ALL_RUNNABLES_WRITING_TO_OR_READING_FROM_DATA_PROTOTYPE_GROUP RunnableEntityGroup of the ConsistencyNeed shall contain all RunnableEntitys with a implicit write or read access to any of the VariableDataPrototypes in the attached DataPrototypeGroup.

EXPLICIT_RUNNABLE_ENTITY_GROUP RunnableEntityGroup of the ConsistencyNeeds shall contain RunnableEntitys according functional requirements

⌋

5.2.11 Blueprinting DataConstr

[TPS_STDT_00016] Blueprinting DataConstr ⌈DataConstr can be blueprinted.⌋

5.2.12 Blueprinting DataTypeMappingSet

[TPS_STDT_00017] Blueprinting DataTypeMappingSet ⌈DataTypeMappingSet can be blueprinted.⌋

5.2.13 Blueprinting EcucDefinitionCollection

[TPS_STDT_00018] Blueprinting EcucDefinitionCollection ⌈EcucDefinitionCollection can be blueprinted.⌋

5.2.14 Blueprinting EcucModuleDef

[TPS_STDT_00019] Blueprinting EcucModuleDef ⌈EcucModuleDef can be blueprinted.⌋

Note that this is intended for company internal use. Please refer to chapter 5.1.7.

5.2.15 Blueprinting FlatMap

[TPS_STDT_00035] Blueprinting FlatMap ⌈FlatMap can be blueprinted.⌋

Usecase for blueprints of FlatMap is given in [16].

5.2.16 Blueprinting ImplementationDataType

[TPS_STDT_00020] Blueprinting ImplementationDataType ⌈ImplementationDataType can be blueprinted.⌋

5.2.17 Blueprinting KeywordSet

[TPS_STDT_00077] Blueprinting KeywordSet ⌈KeywordSet can be blueprinted. The following derivation rules apply:

- No keywords may be removed from or added to the KeywordSet
- The shortName of Keyword shall not be changed or extended
- [TPS_STDT_00085] applies except that longName of Keyword shall not be changed, but it is allowed to add representations in further languages.
- The abbrName shall not be changed or extended(AbbrName)
- The classification of a Keyword shall not be changed but it is allowed to provide additional classification.

⌋

5.2.18 Blueprinting LifeCycleStateDefinitionGroups and LifeCycleStates

[TPS_STDT_00043] Blueprinting LifeCycleStateDefinitionGroup ⌈LifeCycleStateDefinitionGroup and LifeCycleState can be blueprinted. [TPS_STDT_00065] applies such that blueprints of LifeCycleState are aggregated in a blueprint of LifeCycleStateDefinitionGroup.⌋

5.2.19 Blueprinting ModeDeclarationGroup

[TPS_STDT_00031] Blueprinting ModeDeclarationGroup ⌈ModeDeclarationGroup can be blueprinted.⌋

5.2.20 Blueprinting PortPrototype

One of the major activities of the AUTOSAR initiative is the standardization of application interfaces. That is, in terms of the AUTOSAR meta-model the standardization mainly applies to the definition of PortPrototypes for specific purposes.

Due to the structure of the AUTOSAR meta-model it is not possible to merely express a standardized PortPrototype because for good reasons the latter does not exist on its own but is always owned by a SwComponentType.

Therefore, in the past the standardization of "application interfaces" involuntarily also involved the creation of SwComponentTypes. This unnecessary complexity can be overcome by the usage of a PortPrototypeBlueprint.

[TPS_STDT_00007] Blueprinting PortPrototype ⌈PortPrototype can be blueprinted by the specific meta class PortPrototypeBlueprint.⌋

For the mapping of PortPrototypeBlueprints see figure 5.3.

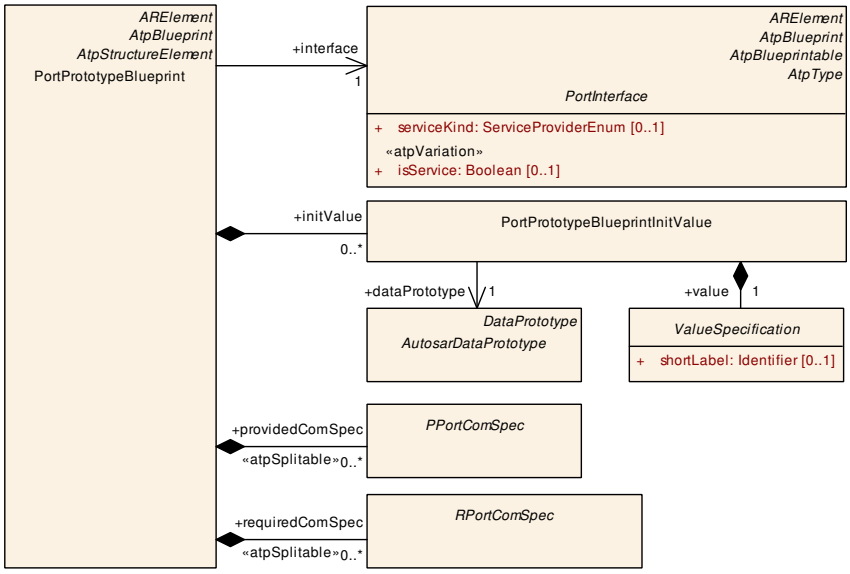


Figure 5.15: Blueprinting Port Prototype

A PortPrototypeBlueprint has the following characteristics:

- It is an ARElement and does therefore not require any element other than an ARPackage as context. It is therefore not necessary to involve "auxiliary" model elements into the definition of a standardized "application interface" for the mere purpose of conforming to the AUTOSAR meta-model.
- It acts as a "blueprint" for the creation of PortPrototypes. That is, probably supported by the used authoring tool, the user picks a specific PortPrototypeBlueprint and creates a PortPrototype out of it. The structure of the created PortPrototype is indistinguishable from a PortPrototype created without taking a PortPrototypeBlueprint as a blueprint. An PortProto-

typeBlueprint can be taken as the blueprint for as many PortPrototypes as required.

- It is possible to define additional attributes that are taken over to the created PortPrototype. For example, in some cases the definition of an initial value[2] is part of the definition of a standardized "application interface". Therefore, PortPrototypeBlueprint also supports the definition of an initValue, which needs to be moved to the appropriate ComSpecs.
- It has a reference to the corresponding PortInterface. If the referenced PortInterface is not a blueprint, it can directly be taken over by the PortPrototype created out of the PortPrototypeBlueprint such that the new PortPrototype references the PortInterface. If the referenced PortInterface is a blueprint, it is necessary to derive a PortInterface and reference this in the PortPrototype.
- It does not make any assumptions whether the PortPrototype created out of it will be a PPortPrototype or an RPortPrototype.
- It can basically be used for all kinds of PortInterfaces, i.e. it is not constrained to e.g. SenderReceiverInterfaces although this kind of PortInterface will most likely get a significant share of the usage of PortPrototypeBlueprint
- It can only be used for the standardization of “application interfaces”. A PortPrototypeBlueprint does not play any role in the formal description of any SwComponentType or related model artifacts (see also [TPS_STDT_00044]).

[TPS_STDT_00061] PortPrototypeBlueprint can own both RPortComSpecs and PPortComSpecs ⌈PortPrototypeBlueprint can own both RPortComSpecs and PPortComSpecs at the same time. The different ComSpecs are applicable for the derived PPortPrototypes, RPortPrototypes and PRPortPrototypes according the given communication direction. The [constr_1043] (PortInterface vs. ComSpec) in Software Component Template ([17]) is also applicable in this context.⌋

[TPS_STDT_00082] Multiple existence of initValue in the context of a PortPrototypeBlueprint ⌈If an initValue exists on the NonqueuedReceiverComSpec or at the NonqueuedSenderComSpec the initValues at PortPrototypeBlueprint shall be ignored.⌋

In this context [TPS_SWCT_01219] needs also be respected for a valid blueprint.

```
<PORT-PROTOTYPE-BLUEPRINT>
  <SHORT-NAME NAME-PATTERN="{anyName}">ALgtOnDoorAtFrntLe</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Acceleration Longitudinal on Door at Front Left</L-4>
  </LONG-NAME>
  <DESC>
```

[2] AUTOSAR does not standardize init values for application interfaces, but it is supported for vendor internal use.

```
      <L-2 L="EN">Longitudinal high-g acceleration measured in front left
         door of vehicle (locking in driving direction)</L-2>
    </DESC>
    <INTERFACE-REF DEST="SENDER-RECEIVER-INTERFACE">/AUTOSAR/AISpecification
       /PortInterfaces_Blueprint/AExtForOccptPedSfty1</INTERFACE-REF>
    <PROVIDED-COM-SPECS>
      <NONQUEUED-SENDER-COM-SPEC>
        <NETWORK-REPRESENTATION>
          <SW-DATA-DEF-PROPS-VARIANTS>
            <SW-DATA-DEF-PROPS-CONDITIONAL>
              <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
                 BaseTypes_Blueprint/uint8</BASE-TYPE-REF>
              <COMPU-METHOD-REF DEST="COMPU-METHOD">/AUTOSAR/Example/
                 CompuMethods_Blueprint/AccelerationOnBus</COMPU-METHOD-REF
                 >
            </SW-DATA-DEF-PROPS-CONDITIONAL>
          </SW-DATA-DEF-PROPS-VARIANTS>
        </NETWORK-REPRESENTATION>
        <INIT-VALUE>
          <APPLICATION-VALUE-SPECIFICATION>
            <CATEGORY>VALUE</CATEGORY>
            <SW-VALUE-CONT>
              <SW-VALUES-PHYS>
                <V>42</V>
              </SW-VALUES-PHYS>
            </SW-VALUE-CONT>
          </APPLICATION-VALUE-SPECIFICATION>
        </INIT-VALUE>
      </NONQUEUED-SENDER-COM-SPEC>
    </PROVIDED-COM-SPECS>
  </PORT-PROTOTYPE-BLUEPRINT>
```

Listing 5.4: PortPrototypeBlueprint with ProvidedComSpecs

Class	**PortPrototypeBlueprint**			
Note	This meta-class represents the ability to express a blueprint of a PortPrototype by referring to a particular PortInterface. This blueprint can then be used as a guidance to create particular PortPrototypes which are defined according to this blueprint. By this it is possible to standardize application interfaces without the need to also standardize software-components with PortPrototypes typed by the standardized Port Interfaces. **Tags:** atp.recommendedPackage=PortPrototypeBlueprints			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Collectable Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	PortPrototypeBlueprint InitValue	*	aggr	This specifies the init values for the dataElements in the particular PortPrototypeBlueprint.
interface	PortInterface	1	ref	This is the interface for which the blueprint is defined. It may be a blueprint itself or a standardized PortInterface

▽

△

Class	**PortPrototypeBlueprint**			
providedCom Spec	PPortComSpec	*	aggr	Provided communication attributes per interface element (data element or operation). **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=providedComSpec.dataElement, providedComSpec.getter, providedComSpec.mode Group, providedComSpec.operation, providedCom Spec.parameter, providedComSpec.setter, providedCom Spec.variable
requiredCom Spec	RPortComSpec	*	aggr	Required communication attributes, one for each interface element. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredComSpec.dataElement, requiredComSpec.getter, requiredComSpec.modeGroup, requiredComSpec.operation, requiredCom Spec.parameter, requiredComSpec.setter, requiredCom Spec.variable

Table 5.9: PortPrototypeBlueprint

Class	**PortPrototypeBlueprintInitValue**			
Note	This meta-class represents the ability to express init values in PortPrototypeBlueprints. These init values act as a kind of blueprint from which for example proper ComSpecs can be derived.			
Base	*ARObject*			
Aggregated by	PortPrototypeBlueprint.initValue			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataPrototype	AutosarDataPrototype	1	ref	This is the data prototype for which the init value applies **Tags:** xml.sequenceOffset=30
value	ValueSpecification	1	aggr	This is the init value for the particular data prototype. **Tags:** xml.sequenceOffset=40

Table 5.10: PortPrototypeBlueprintInitValue

As an AUTOSAR model taken for downstream model handling (e.g. generation of an RTE) requires the usage of complete `PortInterface`s it is necessary to derive an "actual" `PortInterface` out of a blueprinted `PortInterface` defined in the standardization process.

[TPS_STDT_00008] Compatibility of `PortPrototype` with Blueprint ⌈[constr_2526], [constr_2527], [constr_2528] and [constr_2529] apply for the compatibility of `PortPrototype`s and `PortPrototypeBlueprint`s⌋

[constr_2526] `PortInterface` need to be compatible to the blueprints ⌈`PortInterface` shall be compatible to their respective blueprints according to the compatibility rules.⌋

[constr_2527] Blueprints shall live in package of a proper category ⌈As explained in detail in the [2], model artifacts (in this case `PortPrototypeBlueprint` and incompletely specified `PortInterface`s) created for the purpose of becoming blueprints shall reside in an `ARPackage` of category `BLUEPRINT`.⌋

[constr_2528] PortPrototypes shall not refer to blueprints of a PortInterface ⌈A port PortPrototype shall not reference a PortInterface which lives in a package of category BLUEPRINT.⌋

[constr_2529] PortPrototypeBlueprints and derived PortPrototypes shall reference proper PortInterfaces ⌈A PortPrototypeBlueprint may reference a blueprint of PortInterface. According to [constr_2570], a system description shall not contain blueprints. Therefore the reference to the PortInterface may need to be rewritten when a PortPrototype is derived from the blueprint.

In this case the PortInterface referenced by the derived PortPrototype shall be compatible to the PortInterface (which is a blueprint) referenced by the PortPrototypeBlueprint.

According to [constr_2526] this can be ensured if the PortInterface referenced by the PortPrototypeBlueprint is the blueprint of the PortInterface referenced by the respective PortPrototype.⌋

Note that [constr_2529] is obviously also fulfilled if the PortPrototypeBlueprint and the derived PortPrototype reference a STANDARD PortInterface (which lives in a ARPackage of category "STANDARD").

5.2.21 Blueprinting PortInterface

[TPS_STDT_00066] Blueprinting PortInterface ⌈PortInterface can be blueprinted.⌋

[constr_2500] PortInterfaces shall be of same kind ⌈Both objects (PortInterfaces) referenced by a blueprint mapping for port interfaces (represented by BlueprintMapping) shall be of the same kind (e.g. both shall be SenderReceiverInterfaces). In other words both interfaces shall be instances of the same meta class.⌋

Note that [constr_2500] is a special case of [constr_2566].

5.2.22 Blueprinting PortInterfaceMapping and PortInterfaceMappingSet

[TPS_STDT_00009] Blueprinting PortInterfaceMapping and PortInterfaceMappingSet ⌈PortInterfaceMapping can be blueprinted. [TPS_STDT_00065] applies such that the blueprints of PortInterfaceMapping are aggregated in a blueprint of PortInterfaceMappingSet.⌋

The intended use cases for blueprinting PortInterfaceMapping are illustrated by figure 5.16. This diagram shows an PortInterface(Blueprint) (*M*), and two ports typed by PortInterface (*S*) respectively by PortInterface(*R*). (*S*) and (*R*) are

mapped to the blueprint (*M*) by a PortInterfaceMapping(Blueprint) (*SMMap* and *RMMap*). From this, it is possible to

1. derive PortInterfaceMapping (*SRMap*) between (*S* and *R*) which is then derived from two blueprints (*SMMap* and *RMMap*)
2. propose connectors between two components using the interfaces (*S* and *R*)

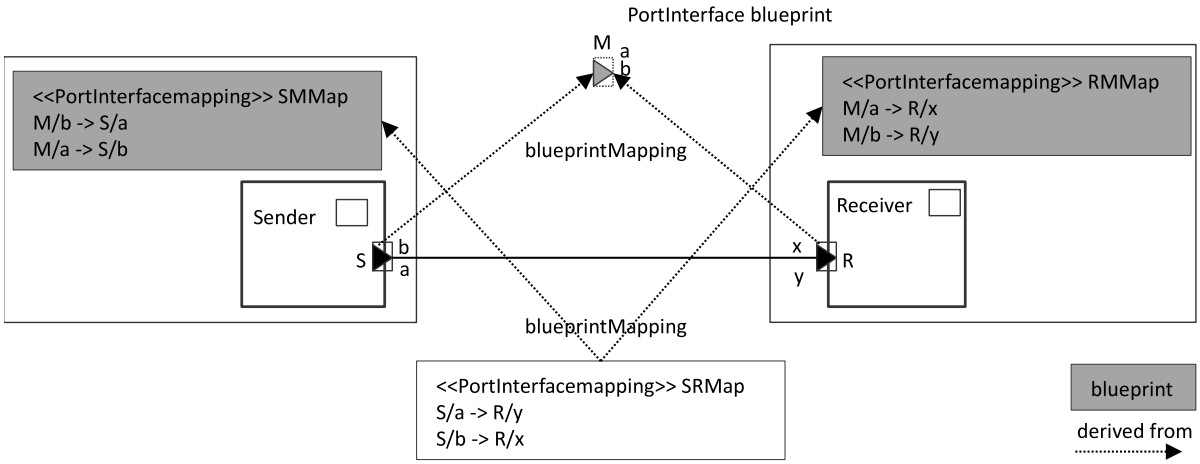


Figure 5.16: Deriving PortInterfaceMapping (1)

The intended derived objects can be determined according to the following steps:

1. find all PortInterface(blueprint)s within the BlueprintMappings of PortInterfaces containing *S* or *R* (in the example it would be *M*)
2. find all PortInterfaceMapping(Blueprint)s containing one of the PortInterface(Blueprint)s from step #1 and one of the PortInterfaces *S* and *R* (in our example it would be *SMMap* and *RMMap*)
3. derive a non blueprint PortInterfaceMapping between *S* and R from the ones found in step #2. Note that all PortInterfaceMappings found so far have a "blueprint reference" and a "non blueprint reference".

   Take one of the PortInterfaceMapping(Blueprint)s from step #2 and replace the "blueprint reference" by the corresponding "non blueprint reference" of the other PortInterfaceMapping(Blueprint)

```
M/b (blueprint in SMMap) -> S/a  <->  M/b (blueprint in RMmap) -> R/y
M/a (blueprint in SMMap) -> S/b  <->  M/a (blueprint in RMmap) -> R/x
```

   For example *M/b* would be substituted by *R/y* and *M/a* by *R/x* resulting in the final mapping ($S/a \rightarrow R/y$, $S/b \rightarrow R/x$).

   Same result is achieved if *M/b* would be substituted by *S/a* and *M/a* by *S/b* resulting in the final mapping ($S/a \rightarrow R/y$, $S/b \rightarrow R/x$).

Implicit mappings (i.e. if data element names between PortInterface and PortInterface(blueprint) are identical then no PortInterfaceMapping(blueprint) is needed) have to be considered too (for example by creating "temporary" mappings).

4. Create BlueprintMappings for the created PortInterfaceMapping (*SRMap*) in step #3 to the involved PortInterfaceMapping(blueprints) (*SMMap* and *RMMap*).

The scenario is shown in these listings:

- Listing A.1 shows the definitions e.g. given by AUTOSAR.
- Listing A.2 shows the part of LeftCompany
- Listing A.3 shows the part of RightCompany
- Listing A.4 shows the part of the integration in a Project

Listing A.2 shows that "LeftCompany" has created the PortInterface named *S* derived from the PortInterface(Blueprint) *M*. Thereby the description **how** this takes place is given in the blueprint of an appropriate PortInterfaceMapping named *SMMap*.

Listing A.3 shows that "RightCompany" has crated the PortInterface named *R* derived from the PortInterface(Blueprint) *M*. Thereby the description **how** this takes place is given in the blueprint of an appropriate PortInterfaceMapping named *RMMap*.

Listing A.4 shows that "Project" used contributions from "RightCompany" and "LeftCompany". Thereby it maps *S* to *R* in PortInterfaceMapping *SRMap*. This is derived from two blueprints (*SMMap* and *SRMap*).

5.2.23 Blueprinting SwBaseType

[TPS_STDT_00022] Blueprinting SwBaseType ⌈SwBaseType can be blueprinted.⌋

5.2.24 Blueprinting SwComponentType

[TPS_STDT_00024] Blueprinting SwComponentType ⌈SwComponentType can be blueprinted.⌋

[constr_2568] SwComponentTypes shall be of same kind ⌈Both objects (SwComponentTypes) referenced by a blueprint mapping for port interfaces (represented by BlueprintMapping) shall be of the same kind (e.g. both shall be AtomicSwComponentTypes). In other words both components shall be instances of the same meta class.⌋

Note that [constr_2568] is a special case of [constr_2566].

5.2.25 Blueprinting SwAddrMethods

[TPS_STDT_00026] Blueprinting `SwAddrMethod` ⌈`SwAddrMethod` can be blueprinted.⌋

5.2.26 Blueprinting VfbTiming

[TPS_STDT_00079] Blueprinting `VfbTiming` ⌈`VfbTiming` can be blueprinted.⌋

One of the essential purposes of blueprinting VFB Timing is enabling one to specify temporal characteristics of interfaces specified in the AUTOSAR Application Interface Table [3]. In particular, one likes to specify timing constraints imposed on sampling rate, recurrence, age, latency, etc. for such interfaces.

Figure 5.17 shows the basic structure of a VFB Timing Blueprint and how the specified timing elements reference other blueprint elements, specifically the elements `PortPrototypeBlueprint` and port interface elements which are referenced by the element `PortInterface`; like variable data prototypes (data elements), client-server operations, mode declarations, and triggers.

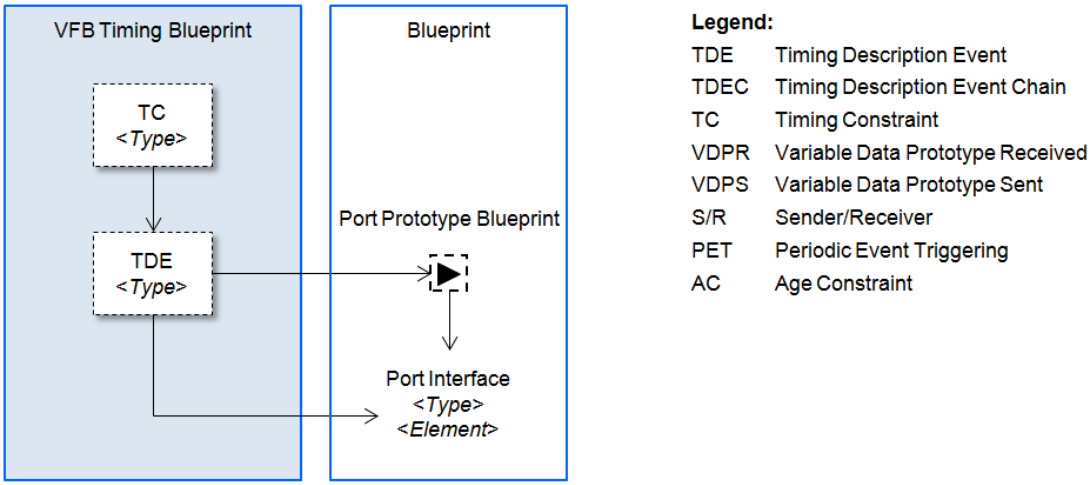


Figure 5.17: VFB Timing Blueprint

A VFB Timing Blueprint consists of timing descriptions events related to the AUTOSAR VFB view, timing description event chains, and timing constraints as defined in the "AUTOSAR Specification of Timing Extensions" [18].

A VFB Timing references the software component it is associated with. In case of a VFB Timing Blueprint this reference need not to be set, but in the derived VFB Timing the `VfbTiming.component` shall be set properly. In addition, any reference to

PortPrototypeBlueprint shall be replaced by the corresponding reference to the PortPrototype.

The following constraints apply to VFB Timing Blueprints and shall be considered when creating such blueprints.

[constr_2589] In VFB Timing Blueprint TDEventVfbPort shall reference PortPrototypeBlueprint ⌈In a VFB Timing Blueprint TDEventVfbPort shall reference PortPrototypeBlueprint. In other words, a VFB Timing Description Event specified in a VFB Timing Blueprint shall always reference a Port Prototype Blueprint.⌋

5.2.26.1 Example

An example for a VFB Timing Blueprint is shown based on [19].

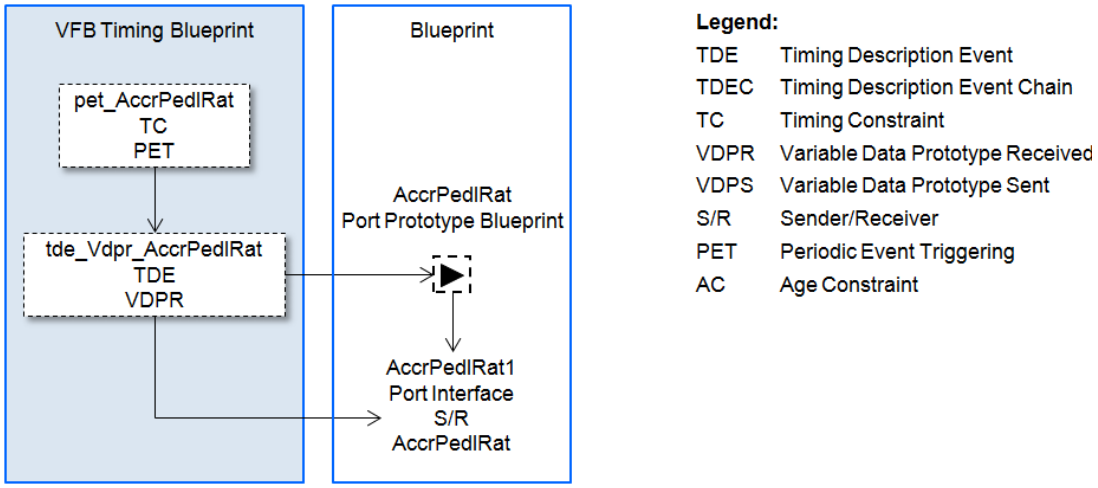
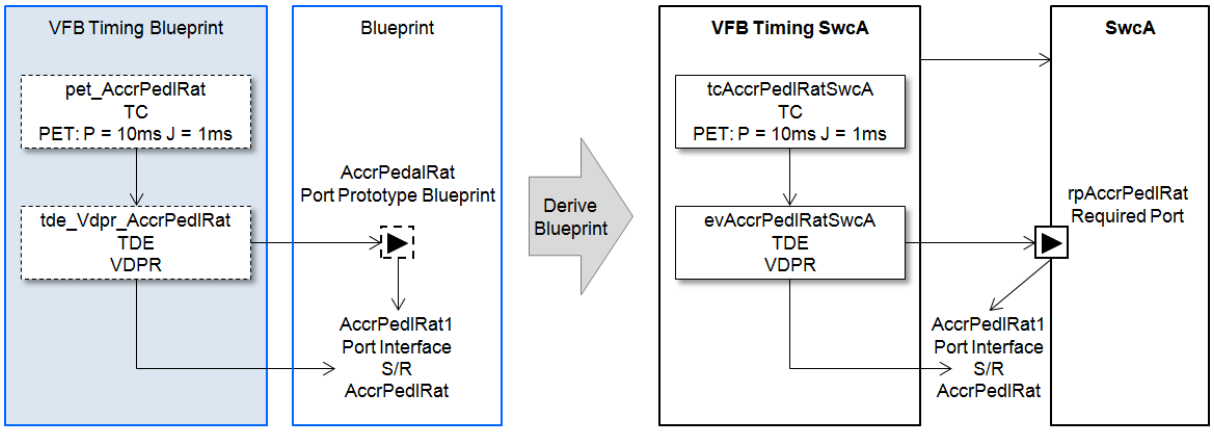


Figure 5.18: VFB Timing Blueprint Simple Example

As sketched in Figure 5.18 a VFB Timing Blueprint is specified. This blueprint consists of a timing description event called “tde_Vdpr_AccrPedlRat” that references the port prototype blueprint called “AccrPedlRat”; and also references the variable data prototype called “AccrPedlRat” of the port interface called “AccrPedlRat1”. The latter is referenced by the mentioned port prototype blueprint, too. In addition, a timing constraint, specifically a periodic event triggering constraint, is imposed on the timing description event. In essence, this timing model specifies that the variable data prototype called “AccrPedlRat” shall be received at a rate given by the periodic event triggering constraint.

The listing A.5 provides the corresponding contents of the ARXML file related to the example shown in Figure 5.18, but contains further timing description events and an additional age timing constraint imposed on the reception of the specific variable data prototype.

Figure 5.19 shows the VFB Timing Blueprint and the derived VFB Timing for a specific software component called “SW-C_A”.



Legend:

TDE	Timing Description Event
TDEC	Timing Description Event Chain
TC	Timing Constraint
VDPR	Variable Data Prototype Received
VDPS	Variable Data Prototype Sent
S/R	Sender/Receiver
PET	Periodic Event Triggering
AC	Age Constraint
P	Period
J	Jitter

Figure 5.19: Deriving a VFB Timing Blueprint

5.2.27 Blueprinting ClientServerInterfaceToBswModuleEntryBlueprintMapping

[TPS_STDT_00083] Blueprinting `ClientServerInterfaceToBswModuleEntryBlueprintMapping` ⌈`ClientServerInterfaceToBswModuleEntryBlueprintMapping` can be blueprinted.⌋

[TPS_STDT_00084] `ClientServerOperationBlueprintMapping` predetermines the implementation of an `ClientServerOperation` ⌈A `ClientServerOperationBlueprintMapping` expresses the intended implementation of a `ClientServerOperation` by a specific `BswModuleEntry` under consideration of the expected usage of `PortDefinedArgumentValue`s.⌋

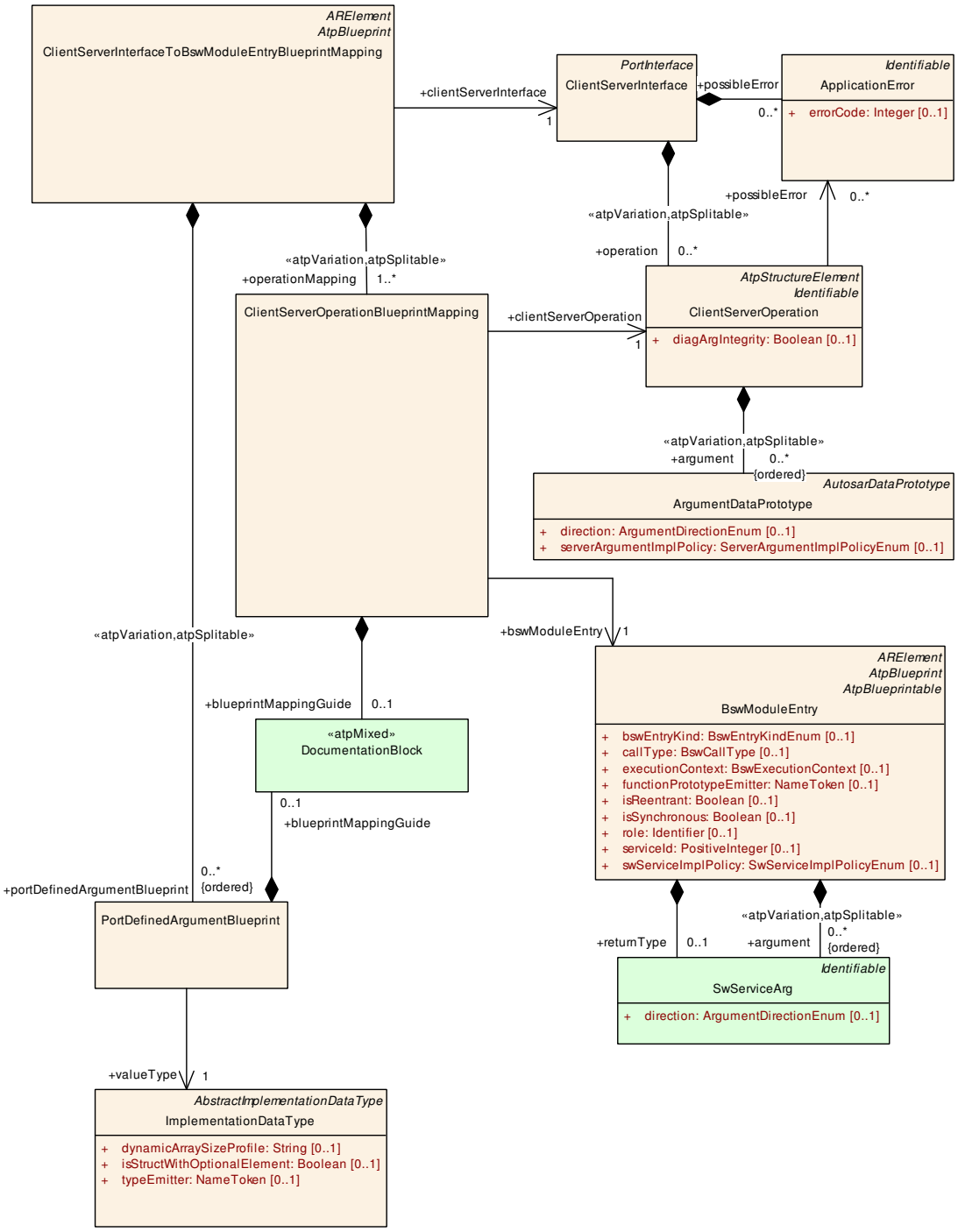


Figure 5.20: Client Server Operation Blueprint Mapping

Class	**ClientServerOperationBlueprintMapping**
Note	This class describes a specific mapping between a ClientServerOperation in a ClientServerInterface blueprint and a BswModuleEntry blueprint.
Base	*ARObject*
Aggregated by	ClientServerInterfaceToBswModuleEntryBlueprintMapping.operationMapping

△

Class	**ClientServerOperationBlueprintMapping**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprint MappingGuide	DocumentationBlock	0..1	aggr	This attribute offers the possibility to provide additional information with respect to the mapping.
bswModule Entry	BswModuleEntry	1	ref	The referenced BswModuleEntry represents the Bsw ModuleEntry the mapping is dedicated to. This Attribute is only used by the AUTOSAR Classic Platform.
clientServer Operation	ClientServerOperation	1	ref	The referenced ClientServerOperation represents the client server operation the mapping is dedicated to.

Table 5.11: ClientServerOperationBlueprintMapping

The ClientServerOperationBlueprintMapping can be used to ensure and/or track the compatibility of BswModuleEntrys which are supposed to implement ClientServerOperations. It can already be defined in an early phase of the methodology when interfaces are defined. Thereby the ClientServerOperationBlueprintMapping can already be defined without all implementation details of the later required SwComponentType, SwcInternalBehavior, BswModuleDescription, BswInternalBehavior and SwcBswMapping.

Please note that the ClientServerInterfaceToBswModuleEntryBlueprintMapping has no direct impact to the later generated RTE. The setup of the RTE is solely determined by the derived objects of ClientServerOperation, BswModuleEntry and the completed software component descriptions and basic software module descriptions respectively.

Such a mapping enables the formal check whether the number of arguments and the data types of arguments of the operation + additional PortDefinedArgumentValues matches the signature of the BswModuleEntry.

[constr_2597] ClientServerOperationBlueprintMapping constrains number of arguments ⌈The number of arguments of the BswModuleEntry referenced by a bswModuleEntry shall be identical to the number of portDefinedArgumentBlueprints of the owning ClientServerInterfaceToBswModuleEntryBlueprintMapping plus the number of ArgumentDataPrototypes aggregated in the role argument of the clientServerOperation⌋

[constr_2598] ClientServerOperationBlueprintMapping constrains the types of arguments ⌈The arguments in the ordered lists bswModuleEntry and the matching arguments in the set union of the ordered lists portDefinedArgumentBlueprint plus clientServerOperation shall result in the identical C data type definitions.⌋

5.3 Deriving from AUTOSAR-provided Blueprints

Model elements provided by AUTOSAR are mainly provided as blueprints. This holds true in particular for the Application Interfaces [3] but also for the Software Specifica-

tions of the BSW layer. These AUTOSAR delivered model elements follow the package structure specified in [TPS_GST_00080].

Figure 5.21 illustrates the methodology to define data types for BSW module. The `BSW Standard Package` contains blueprints. In the above scenario, [TPS_STDT_00067] shall be followed but of course also holds true for the data types of other modules.

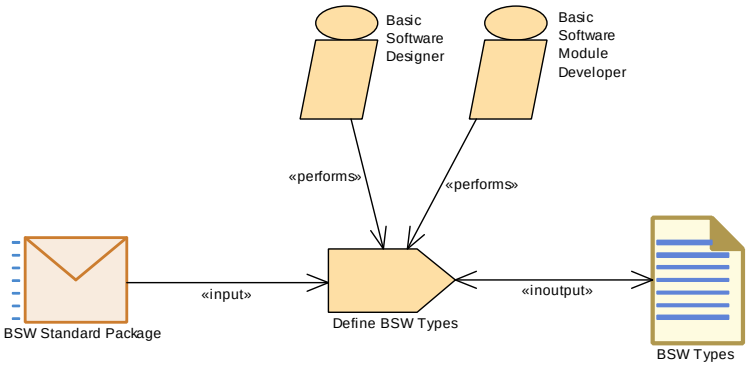


Figure 5.21: Define Bsw Types

[TPS_STDT_00067] Standardized Path for Standardized Elements ⌈Objects derived from standardized blueprints, shall follow a package path as specified in [TPS_GST_00083]. That is, providers of Software components can rely that all AUTOSAR defined model elements can be accessed through a predicable path.⌋

For example the Platform types [20] blueprinted in

`/AUTOSAR/Platform/ImplementationDatatypes_Blueprint/uint8`

shall be implemented in (and therefore safely be accessible through)

`/AUTOSAR_Platform/ImplementationDatatypes/uint8`

6 Keywords

[TPS_STDT_00012] Defining Keywords ⌈The meta-class KeywordSet can be used to define sets of Keywords. The purpose of a Keyword is to contribute parts of names for AUTOSAR model elements.⌋

Keywords are referenced to be part of name pattern as specified in Chapter 5.1.5.

As an example, the shortName "CmftMngt" is composed out of two Keywords with the abbrName "Cmft" and "Mngt".

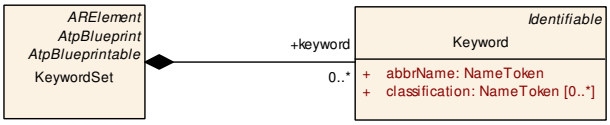


Figure 6.1: Keyword and KeywordSet

[TPS_STDT_00069] Attributes of Keyword ⌈The meta-class Keyword is derived from Identifiable. The attributes of Identifiable shall be applied for Keyword as follows.

shortName represents the unique name of the keyword. In the example above it would be "Cmft". Note that this is used only for identifying the keyword. The contributed name part is taken from abbrName.

longName represents the long form of the keyword, typically its an unabbreviated technical term. In the example above it would be "Comfort".

desc represents the definition of the keyword in terms of a verbal description allowing to identify whether the keyword applies for a specific case. In the example above the description would be "This keyword is used to express something as comfortable or convenient".

introduction represents a verbal description of a use case. This can be used for additional explanations or examples.

⌋

[TPS_STDT_00070] Classification of Keywords ⌈The attribute classification depends on the applied naming convention.⌋

For example, the values could be according to table 2 of [21] such as Action-PhysicalType, Condition-Qualifier, Index, Mean-Environment-Device, Preposition.

Listing A.6 illustrates an example how to use Keyword.

[TPS_STDT_00068] Expressing "stem"-Relation of Keywords ⌈There are keywords which basically stem from the same root. This relationship is expressed by an Collection where the elementRole is named DECLINATION_OF. The root is denoted

sourceElement. The declinations are denoted in element. The root is not a declination of itself, and therefore is not mentioned as an element again.⌋

As an example for [TPS_STDT_00068] the keywords Drvr, Drvg stem from Drv[1]. This is delivered according to the example in Listing A.7

[1]Note that Drv is not an element of this Collection since it is not a declination of itself.

A Examples

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

This chapter contains a collection of selected examples that reflect concepts described in different chapters of this document.

A.1 Example Blueprints

A.1.1 Blueprints of `PortInterfaceMapping`

```
<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaces_Blueprint</SHORT-NAME>
      <CATEGORY>BLUEPRINT</CATEGORY>
      <ELEMENTS>
        <SENDER-RECEIVER-INTERFACE>
          <SHORT-NAME NAME-PATTERN="{anyName}">M</SHORT-NAME>
          <DATA-ELEMENTS>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME NAME-PATTERN="{anyName}">a</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME NAME-PATTERN="{anyName}">b</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
          </DATA-ELEMENTS>
        </SENDER-RECEIVER-INTERFACE>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
```

Listing A.1: Scenario for Blueprints of PortInterfaceMapping (1)

```
<AR-PACKAGE>
  <SHORT-NAME>LeftCompany</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaces</SHORT-NAME>
      <ELEMENTS>
        <SENDER-RECEIVER-INTERFACE>
          <SHORT-NAME>S</SHORT-NAME>
          <DATA-ELEMENTS>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>b</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>a</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
          </DATA-ELEMENTS>
        </SENDER-RECEIVER-INTERFACE>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>BlueprintMappingSets</SHORT-NAME>
      <ELEMENTS>
        <BLUEPRINT-MAPPING-SET>
          <SHORT-NAME>S_isDerivedFrom_M</SHORT-NAME>
          <DESC>
            <L-2 L="EN">This states <E>that</E> S is derived from M</L-2>
          </DESC>
          <BLUEPRINT-MAPS>
            <BLUEPRINT-MAPPING>
              <BLUEPRINT-REF DEST="PORT-INTERFACE">/AUTOSAR/PortInterfaces_Blueprint/M</BLUEPRINT-REF>
              <DERIVED-OBJECT-REF DEST="PORT-INTERFACE">/LeftCompany/PortInterfaces/S</DERIVED-OBJECT-REF>
            </BLUEPRINT-MAPPING>
          </BLUEPRINT-MAPS>
        </BLUEPRINT-MAPPING-SET>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaceMappingSets_Blueprint</SHORT-NAME>
      <CATEGORY>BLUEPRINT</CATEGORY>
      <ELEMENTS>
        <PORT-INTERFACE-MAPPING-SET>
          <SHORT-NAME NAME-PATTERN="{anyName}">BP</SHORT-NAME>
```

```
          <DESC>
            <L-2 L="EN"></L-2>
          </DESC>
          <PORT-INTERFACE-MAPPINGS>
            <VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
              <SHORT-NAME NAME-PATTERN="{anyName}">SMMap</SHORT-NAME>
              <DESC>
                <L-2 L="EN">This defines <E>how</E> S is derived (and therefore mapped to) from M</L-2>
              </DESC>
              <DATA-MAPPINGS>
                <DATA-PROTOTYPE-MAPPING>
                  <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/PortInterfaces_Blueprint/M/a</FIRST-
                      DATA-PROTOTYPE-REF>
                  <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/LeftCompany/PortInterfaces/S/b</SECOND-DATA-
                      PROTOTYPE-REF>
                </DATA-PROTOTYPE-MAPPING>
                <DATA-PROTOTYPE-MAPPING>
                  <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/PortInterfaces_Blueprint/M/b</FIRST-
                      DATA-PROTOTYPE-REF>
                  <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/LeftCompany/PortInterfaces/S/a</SECOND-DATA-
                      PROTOTYPE-REF>
                </DATA-PROTOTYPE-MAPPING>
              </DATA-MAPPINGS>
            </VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
          </PORT-INTERFACE-MAPPINGS>
        </PORT-INTERFACE-MAPPING-SET>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
```

Listing A.2: Scenario for Blueprints of PortInterfaceMapping (2)

```
<AR-PACKAGE>
  <SHORT-NAME>RightCompany</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaces</SHORT-NAME>
      <ELEMENTS>
        <SENDER-RECEIVER-INTERFACE>
          <SHORT-NAME>R</SHORT-NAME>
          <DATA-ELEMENTS>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>x</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>y</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
          </DATA-ELEMENTS>
        </SENDER-RECEIVER-INTERFACE>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>BlueprintMappingSets</SHORT-NAME>
      <ELEMENTS>
        <BLUEPRINT-MAPPING-SET>
          <SHORT-NAME>R_isDerivedFrom_M</SHORT-NAME>
          <DESC>
            <L-2 L="EN">This states <E>that</E> S is derived from M</L-2>
          </DESC>
          <BLUEPRINT-MAPS>
            <BLUEPRINT-MAPPING>
              <BLUEPRINT-REF DEST="PORT-INTERFACE">/AUTOSAR/PortInterfaces_Blueprint/M</BLUEPRINT-REF>
              <DERIVED-OBJECT-REF DEST="PORT-INTERFACE">/RightCompany/PortInterfaces/R</DERIVED-OBJECT-REF>
            </BLUEPRINT-MAPPING>
          </BLUEPRINT-MAPS>
        </BLUEPRINT-MAPPING-SET>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaceMappingSets_Blueprint</SHORT-NAME>
      <CATEGORY>BLUEPRINT</CATEGORY>
      <ELEMENTS>
        <PORT-INTERFACE-MAPPING-SET>
          <SHORT-NAME NAME-PATTERN="{anyName}">BP</SHORT-NAME>
          <PORT-INTERFACE-MAPPINGS>
            <VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
              <SHORT-NAME NAME-PATTERN="{anyName}">RMMap</SHORT-NAME>
              <DESC>
                <L-2 L="EN">This defines <E>how</E> R is derived (and therefore mapped to) from M</L-2>
              </DESC>
              <DATA-MAPPINGS>
                <DATA-PROTOTYPE-MAPPING>
                  <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/PortInterfaces_Blueprint/M/a</FIRST-
                      DATA-PROTOTYPE-REF>
                  <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/RightCompany/PortInterfaces/R/x</SECOND-DATA-
                      PROTOTYPE-REF>
                </DATA-PROTOTYPE-MAPPING>
                <DATA-PROTOTYPE-MAPPING>
                  <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/PortInterfaces_Blueprint/M/b</FIRST-
                      DATA-PROTOTYPE-REF>
                  <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/RightCompany/PortInterfaces/R/y</SECOND-DATA-
                      PROTOTYPE-REF>
                </DATA-PROTOTYPE-MAPPING>
```

```
            </DATA-MAPPINGS>
          </VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
        </PORT-INTERFACE-MAPPINGS>
      </PORT-INTERFACE-MAPPING-SET>
    </ELEMENTS>
  </AR-PACKAGE>
 </AR-PACKAGES>
</AR-PACKAGE>
```

Listing A.3: Scenario for Blueprints of PortInterfaceMapping (3)

```
<AR-PACKAGE>
  <SHORT-NAME>Project</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaceMappingSets</SHORT-NAME>
      <ELEMENTS>
        <PORT-INTERFACE-MAPPING-SET>
          <SHORT-NAME>Set1</SHORT-NAME>
          <PORT-INTERFACE-MAPPINGS>
            <VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
              <SHORT-NAME>SRMap</SHORT-NAME>
              <DESC>
                <L-2 L="EN">This defines <E>how</E> S is mapped R</L-2>
              </DESC>
              <DATA-MAPPINGS>
                <DATA-PROTOTYPE-MAPPING>
                  <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/LeftCompany/PortInterfaces/S/b</FIRST-DATA-
                      PROTOTYPE-REF>
                  <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/RightCompany/PortInterfaces/R/x</SECOND-DATA-
                      PROTOTYPE-REF>
                </DATA-PROTOTYPE-MAPPING>
                <DATA-PROTOTYPE-MAPPING>
                  <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/LeftCompany/PortInterfaces/S/a</FIRST-DATA-
                      PROTOTYPE-REF>
                  <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/RightCompany/PortInterfaces/R/y</SECOND-DATA-
                      PROTOTYPE-REF>
                </DATA-PROTOTYPE-MAPPING>
              </DATA-MAPPINGS>
            </VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
          </PORT-INTERFACE-MAPPINGS>
        </PORT-INTERFACE-MAPPING-SET>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>BlueprintMappingSets</SHORT-NAME>
      <ELEMENTS>
        <BLUEPRINT-MAPPING-SET>
          <SHORT-NAME>ProjectMap1</SHORT-NAME>
          <DESC>
            <L-2 L="EN">This states <E>that</E> SRMap is derived from SMMap and RMMap simultaneously</L-2>
          </DESC>
          <BLUEPRINT-MAPS>
            <BLUEPRINT-MAPPING>
              <BLUEPRINT-REF DEST="PORT-INTERFACE-MAPPING">/LeftCompany/PortInterfaceMappingSets_Blueprint/BP/SMMap</
                  BLUEPRINT-REF>
              <DERIVED-OBJECT-REF DEST="PORT-INTERFACE-MAPPING">/Project/PortInterfaceMappingSets/Set1/SRMap</DERIVED-
                  OBJECT-REF>
            </BLUEPRINT-MAPPING>
            <BLUEPRINT-MAPPING>
              <BLUEPRINT-REF DEST="PORT-INTERFACE-MAPPING">/RightCompany/PortInterfaceMappingSets_Blueprint/BP/RMMap</
                  BLUEPRINT-REF>
              <DERIVED-OBJECT-REF DEST="PORT-INTERFACE-MAPPING">/Project/PortInterfaceMappingSets/Set1/SRMap</DERIVED-
                  OBJECT-REF>
            </BLUEPRINT-MAPPING>
          </BLUEPRINT-MAPS>
        </BLUEPRINT-MAPPING-SET>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
```

Listing A.4: Scenario for Blueprints of PortInterfaceMapping (4)

A.1.2 Blueprints of VfbTiming

```
<AR-PACKAGES>
  <AR-PACKAGE>
    <SHORT-NAME NAME-PATTERN="{anyName}">VfbTimingBlueprint</SHORT-NAME>
    <CATEGORY>BLUEPRINT</CATEGORY>
    <ELEMENTS>
      <VFB-TIMING>
        <SHORT-NAME>vfbTiming_AccrPedlRat</SHORT-NAME>
        <TIMING-DESCRIPTIONS>
          <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>tde_Vdps_AccrPedlRat</SHORT-NAME>
            <IS-EXTERNAL>false</IS-EXTERNAL>
            <PORT-PROTOTYPE-BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT">/AUTOSAR/AISpecification/
                PortPrototypeBlueprints_Blueprint/AccrPedlRat</PORT-PROTOTYPE-BLUEPRINT-REF>
```

```
            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/AISpecification/PortInterfaces_Blueprint/AccrPedlRat1/
                AccrPedlRat</DATA-ELEMENT-REF>
            <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
          </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
          <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>tde_Vdpr_AccrPedlRat</SHORT-NAME>
            <IS-EXTERNAL>false</IS-EXTERNAL>
            <PORT-PROTOTYPE-BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT">/AUTOSAR/AISpecification/
                PortPrototypeBlueprints_Blueprint/AccrPedlRat</PORT-PROTOTYPE-BLUEPRINT-REF>
            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/AISpecification/PortInterfaces_Blueprint/AccrPedlRat1/
                AccrPedlRat</DATA-ELEMENT-REF>
            <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
          </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
          <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>tde_Vdp_AccrPedlRat</SHORT-NAME>
            <IS-EXTERNAL>false</IS-EXTERNAL>
            <PORT-PROTOTYPE-BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT">/AUTOSAR/AISpecification/
                PortPrototypeBlueprints_Blueprint/AccrPedlRat</PORT-PROTOTYPE-BLUEPRINT-REF>
            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/AISpecification/PortInterfaces_Blueprint/AccrPedlRat1/
                AccrPedlRat</DATA-ELEMENT-REF>
          </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
        </TIMING-DESCRIPTIONS>
        <TIMING-REQUIREMENTS>
          <PERIODIC-EVENT-TRIGGERING>
            <SHORT-NAME>pet_AccrPedlRat</SHORT-NAME>
            <EVENT-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/VfbTimingBlueprint/vfbTiming_AccrPedlRat/tde_Vdp_AccrPedlRat
                </EVENT-REF>
            <JITTER>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>1</CSE-CODE-FACTOR>
            </JITTER>
            <PERIOD>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
            </PERIOD>
          </PERIODIC-EVENT-TRIGGERING>
          <AGE-CONSTRAINT>
            <SHORT-NAME>ac_AccrPedlRat</SHORT-NAME>
            <MAXIMUM>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
            </MAXIMUM>
            <SCOPE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/VfbTimingBlueprint/vfbTiming_AccrPedlRat/
                tde_Vdpr_AccrPedlRat</SCOPE-REF>
          </AGE-CONSTRAINT>
        </TIMING-REQUIREMENTS>
      </VFB-TIMING>
    </ELEMENTS>
  </AR-PACKAGE>
</AR-PACKAGES>
```

Listing A.5: Example for VFB Timing Blueprint

A.2 Example Keyword ARXMLs

A.2.1 Example ARXML for Keywords

```
<AR-PACKAGE>
  <SHORT-NAME>KeywordSets</SHORT-NAME>
  <ELEMENTS>
    <KEYWORD-SET>
      <SHORT-NAME>KeywordListComfort</SHORT-NAME>
      <KEYWORDS>
        <KEYWORD>
          <SHORT-NAME>Cmft</SHORT-NAME>
          <LONG-NAME>
            <L-4 L="EN">Comfort</L-4>
          </LONG-NAME>
          <DESC>
            <L-2 L="EN">comfort. this keyword is used to express something as comfortable or convenient</L-2>
          </DESC>
          <ABBR-NAME>Cmft</ABBR-NAME>
          <CLASSIFICATIONS>
            <CLASSIFICATION>Condition-Qualifier</CLASSIFICATION>
          </CLASSIFICATIONS>
        </KEYWORD>
      </KEYWORDS>
    </KEYWORD-SET>
  </ELEMENTS>
</AR-PACKAGE>
```

Listing A.6: example for keywords

A.2.2 Example ARXML for Stem Relation of Keywords

```
<SHORT-NAME>Collections</SHORT-NAME>
```

```
      <ELEMENTS>
        <COLLECTION>
          <SHORT-NAME>Drv_declinations</SHORT-NAME>
          <CATEGORY>RELATION</CATEGORY>
          <COLLECTION-SEMANTICS>DECLINATION_OF</COLLECTION-SEMANTICS>
          <ELEMENT-REFS>
            <ELEMENT-REF BASE="KW" DEST="KEYWORD">KeywordList/Drvr</ELEMENT-REF>
            <ELEMENT-REF BASE="KW" DEST="KEYWORD">KeywordList/Drvg</ELEMENT-REF>
          </ELEMENT-REFS>
          <SOURCE-ELEMENT-REFS>
            <SOURCE-ELEMENT-REF BASE="KW" DEST="KEYWORD">KeywordList/Drv</SOURCE-ELEMENT-REF>
          </SOURCE-ELEMENT-REFS>
        </COLLECTION>
        <COLLECTION>
          <SHORT-NAME>DefinedView</SHORT-NAME>
          <CATEGORY>SET</CATEGORY>
          <AUTO-COLLECT>REF-ALL</AUTO-COLLECT>
          <ELEMENT-ROLE>PART_OF_SUBSET</ELEMENT-ROLE>
          <ELEMENT-REFS>
            <ELEMENT-REF BASE="OPEN" DEST="PORT-PROTOTYPE-BLUEPRINT">EngN</ELEMENT-REF>
          </ELEMENT-REFS>
        </COLLECTION>
        <COLLECTION>
          <SHORT-NAME>ExpandedView</SHORT-NAME>
          <CATEGORY>SET</CATEGORY>
          <AUTO-COLLECT>REF-NONE</AUTO-COLLECT>
          <ELEMENT-ROLE>PART_OF_SUBSET</ELEMENT-ROLE>
          <ELEMENT-REFS>
            <ELEMENT-REF BASE="OPEN" DEST="PORT-PROTOTYPE-BLUEPRINT">EngN</ELEMENT-REF>
            <ELEMENT-REF BASE="OPEN" DEST="PORT-INTERFACE">EngN1</ELEMENT-REF>
            <ELEMENT-REF BASE="OPEN" DEST="APPLICATION-PRIMITIVE-DATA-TYPE">N1</ELEMENT-REF>
            <!-- futher elements are not shown in this example -->
          </ELEMENT-REFS>
        </COLLECTION>
        <COLLECTION>
          <SHORT-NAME>ViewRelation</SHORT-NAME>
          <CATEGORY>RELATION</CATEGORY>
          <ELEMENT-ROLE>AUTO_COLLECTED_FROM</ELEMENT-ROLE>
          <ELEMENT-REFS>
            <ELEMENT-REF BASE="Coll" DEST="COLLECTION">ExpandedView</ELEMENT-REF>
          </ELEMENT-REFS>
          <SOURCE-ELEMENT-REFS>
            <SOURCE-ELEMENT-REF BASE="Coll" DEST="COLLECTION">DefinedView</SOURCE-ELEMENT-REF>
          </SOURCE-ELEMENT-REFS>
        </COLLECTION>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
```

Listing A.7: Example for Stem Relation of Keywords

A.2.3 Example for `BlueprintPolicyNotModifiable`

```
<?xml version="1.0" encoding="utf-8"?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xml="http://www.w3.org/XML/1998/namespace" xmlns:xsi="http://www.w3.org
    /2001/XMLSchema-instance" xsi:schemaLocation="http://autosar.org/schema/r4.0␣AUTOSAR_00054.xsd">
  <ADMIN-DATA>
    <USED-LANGUAGES>
      <L-10 L="EN" xml:space="default">English</L-10>
    </USED-LANGUAGES>
  </ADMIN-DATA>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>CompuMethods_Blueprint</SHORT-NAME>
      <CATEGORY>BLUEPRINT</CATEGORY>
      <ELEMENTS>
        <COMPU-METHOD>
          <SHORT-NAME>ComM_InhibitionStatusType</SHORT-NAME>
          <CATEGORY>BITFIELD_TEXTTABLE</CATEGORY>
          <BLUEPRINT-POLICYS>
            <BLUEPRINT-POLICY-NOT-MODIFIABLE>
              <ATTRIBUTE-NAME>COMPU-INTERNAL-TO-PHYS</ATTRIBUTE-NAME>
            </BLUEPRINT-POLICY-NOT-MODIFIABLE>
          </BLUEPRINT-POLICYS>
          <COMPU-INTERNAL-TO-PHYS>
            <COMPU-SCALES>
              <COMPU-SCALE>
                <SHORT-LABEL>WakeupInhibitionActive</SHORT-LABEL>
                <SYMBOL>WakeupInhibitionActive__FALSE</SYMBOL>
                <DESC>
                  <L-2 L="EN">Bit 0 (LSB): Wake Up inhibition active</L-2>
                </DESC>
                <MASK>0x01</MASK>
                <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-SCALES>
          </COMPU-INTERNAL-TO-PHYS>
```

```
        </COMPU-METHOD>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>
```

Listing A.8: Example for BlueprintPolicyNotModifiable

A.2.4 Example for `BlueprintPolicyList`

```
<?xml version="1.0" encoding="utf-8"?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xml="http://www.w3.org/XML/1998/namespace" xmlns:xsi="http://www.w3.org
    /2001/XMLSchema-instance" xsi:schemaLocation="http://autosar.org/schema/r4.0␣AUTOSAR_00054.xsd">
  <ADMIN-DATA>
    <USED-LANGUAGES>
      <L-10 L="EN" xml:space="default">English</L-10>
    </USED-LANGUAGES>
  </ADMIN-DATA>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>CompuMethods_Blueprint</SHORT-NAME>
      <CATEGORY>BLUEPRINT</CATEGORY>
      <ELEMENTS>
        <COMPU-METHOD>
                  <SHORT-NAME>Dcm_SecLevelType</SHORT-NAME>
                  <CATEGORY>TEXTTABLE</CATEGORY>
                  <BLUEPRINT-POLICYS>
                    <BLUEPRINT-POLICY-LIST>
                      <ATTRIBUTE-NAME>COMPU-INTERNAL-TO-PHYS/COMPU-SCALES/*</ATTRIBUTE-NAME>
                      <BLUEPRINT-DERIVATION-GUIDE>
                        <P>
                          <L-1 L="EN">The range 0x01...0x3F is used configuration dependent</L-1>
                        </P>
                        <P>
                          <L-1 L="EN">The range 0x40...0xFF is reserved by document</L-1>
                        </P>
                      </BLUEPRINT-DERIVATION-GUIDE>
                      <MAX-NUMBER-OF-ELEMENTS BLUEPRINT-VALUE="undefined">undefined</MAX-NUMBER-OF-ELEMENTS>
                      <MIN-NUMBER-OF-ELEMENTS>1</MIN-NUMBER-OF-ELEMENTS>
                    </BLUEPRINT-POLICY-LIST>
                  </BLUEPRINT-POLICYS>
                  <COMPU-INTERNAL-TO-PHYS>
                    <COMPU-SCALES>
                      <COMPU-SCALE>
                        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0x00</LOWER-LIMIT>
                        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0x00</UPPER-LIMIT>
                        <COMPU-CONST>
                          <VT>DCM_SEC_LEV_LOCKED</VT>
                        </COMPU-CONST>
                      </COMPU-SCALE>
                    </COMPU-SCALES>
                  </COMPU-INTERNAL-TO-PHYS>
                </COMPU-METHOD>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>
```

Listing A.9: Example for BlueprintPolicyList

A.2.5 Example for `BlueprintPolicySingle`

The listing A.10 illustrates the use of `BlueprintPolicySingle`.

```
        <AR-PACKAGE>
          <SHORT-NAME>PortPrototypeBlueprints_Blueprint</SHORT-NAME>
                  <CATEGORY>BLUEPRINT</CATEGORY>
                  <ELEMENTS>
            <!-- Use Case POLICY-SINGLE INTERFACE-REF -->
                    <PORT-PROTOTYPE-BLUEPRINT>
                      <SHORT-NAME NAME-PATTERN="{anyName}">AFbForCmft</SHORT-NAME>
                      <LONG-NAME>
                        <L-4 L="EN">Acceleration Feedback for Comfort</L-4>
                      </LONG-NAME>
                      <DESC>
                        <L-2 L="EN">Cluster of information regarding acceleration and acceleration saturation feedbacks from
                            Vehicle Longitudinal Control (VLC) to Adaptive Cruise Control (ACC). This information is used
                           for comfort reasons.</L-2>
                      </DESC>
                      <BLUEPRINT-POLICYS>
                        <BLUEPRINT-POLICY-SINGLE>
                          <ATTRIBUTE-NAME>INTERFACE-REF</ATTRIBUTE-NAME>
                          <BLUEPRINT-DERIVATION-GUIDE>
                            <P>
                              <L-1 L="EN">Shall only refer to an interface of vendor xyz with the same shortname.</L-1>
                            </P>
                          </BLUEPRINT-DERIVATION-GUIDE>
```

```
                    </BLUEPRINT-POLICY-SINGLE>
                  </BLUEPRINT-POLICYS>
                  <INTERFACE-REF DEST="SENDER-RECEIVER-INTERFACE">/AUTOSAR/AISpecification/PortInterfaces_Blueprint/
                     AFbForCmft1</INTERFACE-REF>
               </PORT-PROTOTYPE-BLUEPRINT>
         </ELEMENTS>
      </AR-PACKAGE>
```

Listing A.10: Example for BlueprintPolicySingle

In listing A.11 the `BlueprintPolicySingle` selects an element node with attribute which equals a defined string (PORTS/P-PORT-PROTOTYPE/SHORT-NAME[@NAME-PATTERN='{Name}_AsymDecrypt']).

```
<BLUEPRINT-POLICY-SINGLE>
   <ATTRIBUTE-NAME>PORTS/P-PORT-PROTOTYPE/SHORT-NAME[@NAME-PATTERN='{Name}_AsymDecrypt']</ATTRIBUTE-NAME>
   <BLUEPRINT-DERIVATION-GUIDE>
      <P>
         <L-1 L="EN">Name = {ecuc(Csm/CsmAsymDecrypt/CsmAsymDecryptConfig.SHORT-NAME)}</L-1>
      </P>
   </BLUEPRINT-DERIVATION-GUIDE>
</BLUEPRINT-POLICY-SINGLE>
```

Listing A.11: Example for BlueprintPolicySingle with attribute name pattern

This results in the selection of the element node illustrated in listing A.12.

```
<P-PORT-PROTOTYPE>
   <SHORT-NAME NAME-PATTERN="{Name}_AsymDecrypt">AsymDecrypt</SHORT-NAME>
   <PROVIDED-INTERFACE-TREF DEST="CLIENT-SERVER-INTERFACE">/AUTOSAR/Csm/ClientServerInterfaces_Blueprint/CsmAsymDecrypt</
      PROVIDED-INTERFACE-TREF>
</P-PORT-PROTOTYPE>
```

Listing A.12: Selected element node <SHORT-NAME>

In listing A.13 the `BlueprintPolicySingle` selects an element node which contains a defined text pattern (OPERATIONS/CLIENT-SERVER-OPERATION[SHORT-NAME/text()="ReadData"]/ARGUMENTS/ARGUMENT-DATA-PROTOTYPE[SHORT-NAME/text()="Data"]).

```
<BLUEPRINT-POLICY-SINGLE>
   <ATTRIBUTE-NAME>OPERATIONS/CLIENT-SERVER-OPERATION[SHORT-NAME/text()="ReadData"]/ARGUMENTS/ARGUMENT-DATA-PROTOTYPE[SHORT-NAME
      /text()="Data"]</ATTRIBUTE-NAME>
   <BLUEPRINT-DERIVATION-GUIDE>
      <P>
         <L-1 L="EN">Data = {ecuc(Dem/DemGeneral/DemDataElementClass.SHORT-NAME)}</L-1>
      </P>
   </BLUEPRINT-DERIVATION-GUIDE>
</BLUEPRINT-POLICY-SINGLE>
```

Listing A.13: Example for BlueprintPolicySingle with text pattern

This results in the selection of the element node (ARGUMENTS/ARGUMENT-DATA-PROTOTYPE/SHORT-NAME) with SHORT-NAME equal to 'Data' in case (CLIENT-SERVER-OPERATION/SHORT-NAME) is equal to 'ReadData', see listing A.14.

```
<OPERATIONS>
   <CLIENT-SERVER-OPERATION>
      <SHORT-NAME>ReadData</SHORT-NAME>
      <INTRODUCTION>
         <P>
            <L-1 L="EN">The server is not allowed to return E_NOT_OK, but shall always provide a valid data value (e.g. a default/
               replacement value in an error-case) to Dcm/Dem nevertheless the signature of the operation includes E_NOT_OK to
               ensure compatibility between server runnable and RTE Call API, since the RTE may return negative Std_Return values
               in certain cases (e.g. partition of server stopped)</L-1>
         </P>
      </INTRODUCTION>
      <ARGUMENTS>
         <ARGUMENT-DATA-PROTOTYPE>
            <SHORT-NAME>Data</SHORT-NAME>
            <TYPE-TREF DEST="IMPLEMENTATION-DATA-TYPE">/AUTOSAR/Dem/ImplementationDataTypes_Blueprint/DataArrayType</TYPE-TREF>
            <DIRECTION>OUT</DIRECTION>
         </ARGUMENT-DATA-PROTOTYPE>
      </ARGUMENTS>
      <POSSIBLE-ERROR-REFS>
```

```
        <POSSIBLE-ERROR-REF DEST="APPLICATION-ERROR">/AUTOSAR/Dem/ClientServerInterfaces_Blueprint/DataServices/E_OK</POSSIBLE-
            ERROR-REF>
        <POSSIBLE-ERROR-REF DEST="APPLICATION-ERROR">/AUTOSAR/Dem/ClientServerInterfaces_Blueprint/DataServices/E_NOT_OK</POSSIBLE
            -ERROR-REF>
      </POSSIBLE-ERROR-REFS>
    </CLIENT-SERVER-OPERATION>
</OPERATIONS>
```

Listing A.14: Example for BlueprintPolicySingle with text pattern

B Mentioned Class Tables

This chapter contains the remaining set of meta-class tables which are not shown directly in the main body of this document.

Class	***ARElement*** (abstract)			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, ApplicabilityInfoSet, Application Partition, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BswEntryRelationshipSet, BswModule Description, BswModuleEntry, BuildActionManifest, CalibrationParameterValueSet, ClientIdDefinitionSet, ClientServerInterfaceToBswModuleEntryBlueprintMapping, Collection, CompuMethod, Consistency NeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CpSoftwareCluster, Cp SoftwareClusterBinaryManifestDescriptor, CpSoftwareClusterMappingSet, CpSoftwareClusterResource Pool, CryptoEllipticCurveProps, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, CryptoSignatureScheme, DataConstr, DataTransformationSet, DataTypeMapping Set, DdsCpConfig, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, Dlt ArgumentPropsSet, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, EcucDefinition Collection, EcucDestinationUriDefSet, EcucModuleConfigurationValues, EcucModuleDef, EcucValue Collection, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FirewallRule, FlatMap, GeneralPurpose Connection, HwCategory, HwElement, HwType, *IEEE1722TpConnection*, IPSecConfigProps, IPv6Ext HeaderFilterSet, *IdsCommonElement*, IdsDesign, *Implementation*, ImpositionTimeDefinitionGroup, InterpolationRoutineMappingSet, J1939ControllerApplication, KeywordSet, LifeCycleInfoSet, LifeCycle StateDefinitionGroup, LogAndTraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipant Set, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, OsTaskProxy, PhysicalDimension, PhysicalDimensionMappingSet, *PortInterface*, PortInterfaceMappingSet, Port PrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, RapidPrototypingScenario, SdgDef, *SecureComProps*, SignalServiceTranslationPropsSet, SomeipSd ClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroup TimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, SwComponent MappingConstraints, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValue Set, SwcBswMapping, System, SystemComSpecDefinitionSet, SystemSignal, SystemSignalGroup, TDCpSoftwareClusterMappingSet, TcpOptionFilterSet, *TimingExtension*, TlsConnectionGroup, TlvData IdDefinitionSet, TransformationPropsSet, Unit, UnitGroup, *UploadablePackageElement*, ViewMapSet			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.1: ARElement

Class	**ARPackage**			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ARPackage.arPackage, AUTOSAR.arPackage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ARPackage**			
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arPackage.shortName, arPackage.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element.shortName, element.variation Point.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20
referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=referenceBase.shortLabel xml.sequenceOffset=10

Table B.2: ARPackage

Class	**AUTOSAR**			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. **Tags:** xml.globalElement=true			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=10
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arPackage.shortName, arPackage.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
fileInfo Comment	FileInfoComment	0..1	aggr	This represents a possibility to provide a structured comment in an AUTOSAR file. **Stereotypes:** atpStructuredComment **Tags:** xml.roleElement=true xml.sequenceOffset=-10 xml.typeElement=false
introduction	DocumentationBlock	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes. **Tags:** xml.sequenceOffset=20

Table B.3: AUTOSAR

Class	***AbstractVariationRestriction*** (abstract)			
Note	Defines constraints on the usage of variation and on the valid binding times.			
Base	*ARObject*			
Subclasses	SdgAggregationWithVariation, SdgForeignReferenceWithVariation, SdgPrimitiveAttributeWithVariation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
validBinding Time	FullBindingTimeEnum	*	attr	List of valid binding times. **Tags:** xml.sequenceOffset=20
variation	Boolean	0..1	attr	Defines if the AUTOSAR model may define a Variation Point at this location. **Tags:** xml.sequenceOffset=10

Table B.4: AbstractVariationRestriction

Class	**AclObjectSet**			
Note	This meta class represents the ability to denote a set of objects for which roles and rights (access control lists) shall be defined. It basically can define the objects based on • the nature of objects • the involved blueprints • the artifact in which the objects are serialized • the definition of the object (in a definition - value pattern) • individual reference objects **Tags:** atp.recommendedPackage=AclObjectSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aclObjectClass	ReferrableSubtypes Enum	*	attr	This specifies that the considered objects as instances of the denoted meta class.
aclScope	AclScopeEnum	1	attr	this indicates the scope of the referenced objects.
collection	Collection	0..1	ref	This indicates that the relevant objects are specified via a collection.
derivedFrom Blueprint	AtpBlueprint	*	ref	This association indicates that the considered objects are the ones being derived from the associated blueprint. **Stereotypes:** atpUriDef
engineering Object	AutosarEngineering Object	*	aggr	This indicates an engineering object. The AclPermission relates to all objects in this partial model. This also implies that the other objects in this set shall be placed in the specified engineering object. Note that semantic constraints apply with respect to <<atpSplitable>>
object	Referrable	*	ref	This association applies a particular (usually small) set of objects (e.g. a singular package). Main usage is, if one does not want to create a collection specifically for access control.
objectDefinition	AtpDefinition	*	ref	This denotes an object by its definition. For example the right to manipulate the value of a particular ecuc parameter is denoted by reference to the definition of the parameter. Note that this can also be a reference to a Standard Module Definition. Therefore it is stereotyped by atpUri Def. **Stereotypes:** atpUriDef

Table B.5: AclObjectSet

Class	AclOperation			
Note	This meta class represents the ability to denote a particular operation which may be performed on objects in an AUTOSAR model. **Tags:** atp.recommendedPackage=AclOperations			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
implied Operation	AclOperation	*	ref	This indicates that the related operations are also implied. Therefore the permission is also granted for this operation.

Table B.6: AclOperation

Class	AclPermission			
Note	This meta class represents the ability to represent permissions granted on objects in an AUTOSAR model. **Tags:** atp.recommendedPackage=AclPermissions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
aclContext	NameToken	*	attr	This attribute is intended to specify the context under which the AclPemission is applicable. The values are subject to mutual agreement between the involved stakeholders. For examples the values can be the names of binding times.
aclObject	AclObjectSet	*	ref	This denotes an object to which the AclPermission applies.
aclOperation	AclOperation	*	ref	This denotes an operation which is granted by the given AclPermission.
aclRole	AclRole	*	ref	This denotes the role (individual or even organization) for which the AclPermission. is granted.
aclScope	AclScopeEnum	1	attr	This indicates the scope of applied permissions: explicit, descendant, dependent;

Table B.7: AclPermission

Class	AclRole			
Note	This meta class represents the ability to specify a particular role which is used to grant access rights to AUTOSAR model. The purpose of this meta-class is to support the mutual agreements between the involved parties. **Tags:** atp.recommendedPackage=AclRoles			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ldapUrl	UriString	0..1	attr	This is an URL which allows to represent users or organizations taking the particular role.

Table B.8: AclRole

Class	**AliasNameSet**			
Note	This meta-class represents a set of AliasNames. The AliasNameSet can for example be an input to the A2L-Generator. **Tags:** atp.recommendedPackage=AliasNameSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aliasName	AliasNameAssignment	*	aggr	AliasNames contained in the AliasNameSet. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=aliasName.shortLabel, aliasName.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.9: AliasNameSet

Class	***ApplicationDataType*** (abstract)			
Note	`ApplicationDataType` defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An `ApplicationDataType` represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc. It should be possible to model the application level aspects of a VFB system by using `ApplicationDataType`s only.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*ApplicationCompositeDataType*, ApplicationPrimitiveDataType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.10: ApplicationDataType

Class	**ApplicationSwComponentType**			
Note	The `ApplicationSwComponentType` is used to represent the application software. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.11: ApplicationSwComponentType

Class	**ArgumentDataPrototype**			
Note	An argument of an operation, carries direction and implementation information.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ClientServerOperation.argument			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	ArgumentDirectionEnum	0..1	attr	This attribute specifies the direction of the argument.

▽

△

Class	**ArgumentDataPrototype**			
serverArgument ImplPolicy	ServerArgumentImpl PolicyEnum	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 B.12: ArgumentDataPrototype

Class	***AtomicSwComponentType*** (abstract)			
Note	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Subclasses	ApplicationSwComponentType, ComplexDeviceDriverSwComponentType, EcuAbstractionSwComponent Type, NvBlockSwComponentType, SensorActuatorSwComponentType, ServiceProxySwComponent Type, ServiceSwComponentType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
internalBehavior	SwcInternalBehavior	0..1	aggr	The `SwcInternalBehavior`s owned by an `AtomicSwComponentType` can be located in a different physical file. Therefore the aggregation is <<atp Splitable>>. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior.shortName, internal Behavior.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the `SymbolProps` for the `AtomicSwComponentType`. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=symbolProps.shortName

Table B.13: AtomicSwComponentType

Class	***AtpBlueprint*** (abstract)			
Note	This meta-class represents the ability to act as a Blueprint. As this class is an abstract one, particular blueprint meta-classes inherit from this one.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ARPackage, *AbstractImplementationDataType*, AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, *ApplicationDataType*, BswEntryRelationshipSet, BswModuleDescription, BswModule Entry, *BuildActionEntity*, BuildActionEnvironment, BuildActionManifest, ClientServerInterfaceToBsw ModuleEntryBlueprintMapping, CompuMethod, ConsistencyNeeds, DataConstr, DataTypeMappingSet, EcucDefinitionCollection, EcucDestinationUriDefSet, EcucModuleDef, FlatMap, ImpositionTime, ImpositionTimeDefinitionGroup, KeywordSet, LifeCycleState, LifeCycleStateDefinitionGroup, Mode DeclarationGroup, *PortInterface*, *PortInterfaceMapping*, PortInterfaceMappingSet, PortPrototype Blueprint, SecurityEventContextDataElement, SwAddrMethod, SwBaseType, *SwComponentType*, Vfb Timing			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprintPolicy	BlueprintPolicy	*	aggr	This role indicates whether the blueprintable element will be modifiable or not modifiable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=blueprintPolicy.attributeName

Table B.14: AtpBlueprint

Class	***AtpBlueprintMapping*** (abstract)			
Note	This meta-class represents the ability to express a particular mapping between a blueprint and an element derived from this blueprint. Particular mappings are defined by specializations of this meta-class.			
Base	*ARObject*			
Subclasses	BlueprintMapping			
Aggregated by	BlueprintMappingSet.blueprintMap			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
atpBlueprint	AtpBlueprint	1	ref	This represents the blueprint. **Stereotypes:** atpAbstract; atpUriDef **Tags:** xml.sequenceOffset=50
atpBlueprinted Element	AtpBlueprintable	1	ref	This represents the bluprinted elements which shall be mapped to the blueprint. **Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=60

Table B.15: AtpBlueprintMapping

Class	***AtpBlueprintable*** (abstract)			
Note	This meta-class represents the ability to be derived from a Blueprint. As this class is an abstract one, particular blueprintable meta-classes inherit from this one.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ARPackage, *AbstractImplementationDataType*, AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, *ApplicationDataType*, BswEntryRelationshipSet, BswModuleDescription, BswModule Entry, *BuildActionEntity*, BuildActionEnvironment, BuildActionManifest, CompuMethod, Consistency Needs, DataConstr, DataTypeMappingSet, EcucDefinitionCollection, EcucDestinationUriDefSet, Ecuc ModuleDef, FlatMap, ImpositionTime, ImpositionTimeDefinitionGroup, KeywordSet, LifeCycleState, Life CycleStateDefinitionGroup, ModeDeclarationGroup, *PortInterface*, *PortInterfaceMapping*, PortInterface MappingSet, *PortPrototype*, SecurityEventContextDataElement, SwAddrMethod, SwBaseType, *Sw ComponentType*, VfbTiming			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.16: AtpBlueprintable

Enumeration	**BindingTimeEnum**
Note	This enumerator specifies the applicable binding times for the pre build variation points.
Aggregated by	ConditionByFormula.bindingTime, FMFeature.maximumIntendedBindingTime, FMFeature.minimum IntendedBindingTime, FMFeatureSelection.maximumSelectedBindingTime, FMFeatureSelection. minimumSelectedBindingTime
Literal	***Description***
codeGeneration Time	• Coding by hand, based on requirements document. • Tool based code generation, e.g. from a model. • The model may contain variants. • Only code for the selected variant(s) is actually generated. **Tags:** atp.EnumerationLiteralIndex=0
linkTime	Configure what is included in object code, and what is omitted Based on which variant(s) are selected E.g. for modules that are delivered as object code (as opposed to those that are delivered as source code) **Tags:** atp.EnumerationLiteralIndex=1

▽

△

Enumeration	**BindingTimeEnum**
preCompileTime	This is typically the C-Preprocessor. Exclude parts of the code from the compilation process, e.g., because they are not required for the selected variant, because they are incompatible with the selected variant, because they require resources that are not present in the selected variant. Object code is only generated for the selected variant(s). The code that is excluded at this stage code will not be available at later stages. **Tags:** atp.EnumerationLiteralIndex=2
systemDesignTime	• Designing the VFB. • Software Component types (PortInterfaces). • SWC Prototypes and the Connections between SWCprototypes. • Designing the Topology • ECUs and interconnecting Networks • Designing the Communication Matrix and Data Mapping **Tags:** atp.EnumerationLiteralIndex=3

Table B.17: BindingTimeEnum

Class	«atpMixedString» **BlueprintFormula**			
Note	This class express the extension of the Formula Language to provide formalized blueprint-Value resp. blueprintCondition.			
Base	*ARObject*, *FormulaExpression*, *SwSystemconstDependentFormula*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
ecuc	EcucDefinitionElement	1	ref	The EcucDefinitionElement serves as a argument for the formular. This Attribute is only used by the AUTOSAR Classic Platform.
verbatim	MultiLanguageVerbatim	1	aggr	This represents an informal term in the expression as verbatim text. Note that the result of this is same as formula keyword "undefined".

Table B.18: BlueprintFormula

Class	**BlueprintMapping**			
Note	This meta-class represents the ability to map two an object and its blueprint.			
Base	*ARObject*, *AtpBlueprintMapping*			
Aggregated by	BlueprintMappingSet.blueprintMap			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
blueprint	AtpBlueprint	1	ref	This represents the mapped blueprint. **Stereotypes:** atpIdentityContributor
derivedObject	AtpBlueprintable	1	ref	This represents the object which was derived from the blueprint. **Stereotypes:** atpIdentityContributor

Table B.19: BlueprintMapping

Class	***BlueprintPolicy*** (abstract)
Note	This meta-class represents the ability to indicate whether blueprintable elements will be modifiable or not modifiable.
Base	*ARObject*
Subclasses	*BlueprintPolicyModifiable*, BlueprintPolicyNotModifiable

▽

△

Class	***BlueprintPolicy*** (abstract)			
Aggregated by	*AtpBlueprint*.blueprintPolicy			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
attributeName	String	1	attr	This identifies the related attribute of a BlueprintPolicy. For navigation over the model a subset of xpath expressions is used. **Stereotypes:** atpIdentityContributor

Table B.20: BlueprintPolicy

Class	**BlueprintPolicyList**			
Note	The class represents that the related attribute is modifiable during the blueprinting. It applies only to attribute with upper multiplicity greater than 1.			
Base	*ARObject*, *BlueprintPolicy*, *BlueprintPolicyModifiable*			
Aggregated by	*AtpBlueprint*.blueprintPolicy			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxNumberOf Elements	PositiveInteger	1	attr	Maximum number of elements in list. If the maximum number is not constraint it shall be set to "undefined". **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
minNumberOf Elements	PositiveInteger	1	attr	Minimum number of elements in the list. If the minimum number is not constraint it shall be set to "undefined". **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime

Table B.21: BlueprintPolicyList

Class	**BlueprintPolicyNotModifiable**			
Note	The class represents that the related attribute is not modifiable during the blueprinting.			
Base	*ARObject*, *BlueprintPolicy*			
Aggregated by	*AtpBlueprint*.blueprintPolicy			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.22: BlueprintPolicyNotModifiable

Class	**BlueprintPolicySingle**			
Note	The class represents that the related attribute is modifiable during the blueprinting. It applies only to attribute with upper multiplicity equal 1.			
Base	*ARObject*, *BlueprintPolicy*, *BlueprintPolicyModifiable*			
Aggregated by	*AtpBlueprint*.blueprintPolicy			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.23: BlueprintPolicySingle

Class	**BswInternalBehavior**
Note	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 BswInternalBehaviors referring to the same BswModule Description.

▽

△

Class	BswInternalBehavior			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *InternalBehavior*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.internalBehavior			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arTypedPerInstanceMemory	VariableDataPrototype	*	aggr	Defines an AUTOSAR typed memory-block that needs to be available for each instance of the Basic Software Module. The aggregation of arTypedPerInstanceMemory 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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arTypedPerInstanceMemory.shortName, arTypedPerInstanceMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
bswPerInstanceMemoryPolicy	BswPerInstanceMemoryPolicy	*	aggr	Policy for a arTypedPerInstanceMemory The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswPerInstanceMemoryPolicy, bswPerInstanceMemoryPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
clientPolicy	BswClientPolicy	*	aggr	Policy for a requiredClientServerEntry. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=clientPolicy, clientPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
distinguishedPartition	BswDistinguishedPartition	*	aggr	Indicates an abstract partition context in which the enclosing BswModuleEntity can be executed. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=distinguishedPartition.shortName, distinguishedPartition.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
entity	BswModuleEntity	*	aggr	A code entity for which the behavior is described **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=entity.shortName, entity.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=5
event	BswEvent	*	aggr	An event required by this module behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=event.shortName, event.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=10

▽

△

Class	**BswInternalBehavior**			
exclusiveArea Policy	BswExclusiveArea Policy	*	aggr	Policy for an ExclusiveArea in this BswInternalBehavior. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaPolicy, exclusiveArea Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedData TypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a basic software module for its implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedDataTypeSet
includedMode Declaration GroupSet	IncludedMode DeclarationGroupSet	*	aggr	This aggregation represents the included Mode DeclarationGroups **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedModeDeclarationGroupSet
internal TriggeringPoint	BswInternalTriggering Point	*	aggr	An internal triggering point. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPoint.shortName, internal TriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=2 This Attribute is only used by the AUTOSAR Classic Platform.
internal TriggeringPoint Policy	BswInternalTriggering PointPolicy	*	aggr	Policy for an internalTriggeringPoint in this BswInternal Behavior.. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPointPolicy, internal TriggeringPointPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeReceiver Policy	BswModeReceiver Policy	*	aggr	Implementation policy for the reception of mode switches. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeReceiverPolicy, modeReceiver Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25
modeSender Policy	BswModeSenderPolicy	*	aggr	Implementation policy for providing a mode group. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeSenderPolicy, modeSender Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
parameterPolicy	BswParameterPolicy	*	aggr	Policy for a perInstanceParameter in this BswInternal Behavior. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterPolicy, parameterPolicy.variation Point.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**BswInternalBehavior**			
perInstance Parameter	ParameterData Prototype	*	aggr	Describes a read only memory object containing characteristic value(s) needed by this BswInternal Behavior. The role name perInstanceParameter is chosen in analogy to the similar role in the context of SwcInternal Behavior. 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. The aggregation is subject to variability with the purpose to support implementation variants. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceParameter.shortName, per InstanceParameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
receptionPolicy	BswDataReception Policy	*	aggr	Data reception policy for inter-partition and/or inter-core communication. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=receptionPolicy, receptionPolicy.variation Point.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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=releasedTriggerPolicy, releasedTrigger Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
schedulerName Prefix	BswSchedulerName Prefix	*	aggr	Optional definition of one or more prefixes to be used for the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=schedulerNamePrefix.shortName, scheduler NamePrefix.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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sendPolicy, sendPolicy.variationPoint.short Label vh.latestBindingTime=preCompileTime
service Dependency	BswService Dependency	*	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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceDependency.ident.shortName, serviceDependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

▽

△

Class	**BswInternalBehavior**			
triggerDirect Implementation	BswTriggerDirect Implementation	*	aggr	Specifies a trigger to be directly implemented via OS calls. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=triggerDirectImplementation, triggerDirect Implementation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=15
variationPoint Proxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=variationPointProxy.shortName

Table B.24: BswInternalBehavior

Class	**BswModuleDescription**			
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. **Tags:** atp.recommendedPackage=BswModuleDescriptions This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswModule Dependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDependency.shortName, bsw ModuleDependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
bswModule Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDocumentation, bswModule Documentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6
expectedEntry	BswModuleEntry	*	ref	Indicates an entry which is required by this module. Replacement of outgoingCallback / requiredEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=expectedEntry.bswModuleEntry, expected Entry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	*	ref	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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implementedEntry.bswModuleEntry, implementedEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**BswModuleDescription**			
internalBehavior	BswInternalBehavior	*	aggr	The various BswInternalBehaviors associated with a Bsw ModuleDescription can be distributed over several physical files. Therefore the aggregation is <<atp Splitable>>. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=internalBehavior.shortName xml.sequenceOffset=65
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. **Tags:** xml.sequenceOffset=5
providedClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module provides a client server entry which can be called from another partition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedClientServerEntry.shortName, providedClientServerEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedData.shortName, provided Data.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
providedMode Group	ModeDeclarationGroup Prototype	*	aggr	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, EcuAbstraction SwComponentType or ComplexDeviceDriverSw ComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedModeGroup.shortName, provided ModeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25

▽

△

Class	**BswModuleDescription**			
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, Ecu AbstractionSwComponentType or ComplexDeviceDriver SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=releasedTrigger.shortName, released Trigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another partition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredClientServerEntry.shortName, requiredClientServerEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder 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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredData.shortName, required Data.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
requiredMode Group	ModeDeclarationGroup Prototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredModeGroup.shortName, required ModeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredTrigger.shortName, required Trigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table B.25: BswModuleDescription

Class	BswModuleEntry			
Note	This class represents a single API entry (C-function prototype) into the BSW module or cluster. The name of the C-function is equal to the short name of this element with one exception: In case of multiple instances of a module on the same CPU, special rules for "infixes" apply, see description of class BswImplementation. **Tags:** atp.recommendedPackage=BswModuleEntrys This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
argument (ordered)	SwServiceArg	*	aggr	An argument belonging to this BswModuleEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=argument.shortName, argument.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=45
bswEntryKind	BswEntryKindEnum	0..1	attr	This describes whether the entry is concrete or abstract. If the attribute is missing the entry is considered as concrete. **Tags:** xml.sequenceOffset=40
callType	BswCallType	0..1	attr	The type of call associated with this service. **Tags:** xml.sequenceOffset=25
executionContext	BswExecutionContext	0..1	attr	Specifies the execution context which is required (in case of entries into this module) or guaranteed (in case of entries called from this module) for this service. **Tags:** xml.sequenceOffset=30
functionPrototypeEmitter	NameToken	0..1	attr	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.
isReentrant	Boolean	0..1	attr	Reentrancy from the viewpoint of function callers: • true: Enables the service to be invoked again, before the service has finished. • false: It is prohibited to invoke the service again before is has finished. **Tags:** xml.sequenceOffset=15
isSynchronous	Boolean	0..1	attr	Synchronicity from the viewpoint of function callers: • true: This calls a synchronous service, i.e. the service is completed when the call returns. • false: The service (on semantical level) may not be complete when the call returns. **Tags:** xml.sequenceOffset=20
returnType	SwServiceArg	0..1	aggr	The return type belonging to this bswModuleEntry. **Tags:** xml.sequenceOffset=40
role	Identifier	0..1	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
serviceId	PositiveInteger	0..1	attr	Refers to the service identifier of the Standardized Interfaces of AUTOSAR basic software. For non-standardized interfaces, it can optionally be used for proprietary identification. **Tags:** xml.sequenceOffset=5

△

Class	BswModuleEntry			
swServiceImpl Policy	SwServiceImplPolicy Enum	0..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 B.26: BswModuleEntry

Class	BuildAction			
Note	This meta-class represents the ability to specify a build action.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *BuildActionEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BuildActionManifest.buildAction			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
createdData	BuildActionIoElement	*	aggr	This represents the artifacts which are created by the processor. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=createdData
followUpAction	BuildAction	*	ref	This association specifies a set of follow up actions. **Tags:** xml.sequenceOffset=-80
inputData	BuildActionIoElement	*	aggr	This represents the artifacts which are read by the processor. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=inputData
modifiedData	BuildActionIoElement	*	aggr	This denotes the data which are modified by the action. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=modifiedData
predecessor Action	BuildAction	*	ref	This association specifies a set of predecessors. These actions shall be finished before but necessarily immediately after the given action.. These actions need to be performed in the specified order. **Tags:** xml.sequenceOffset=-90
required Environment	BuildActionEnvironment	1	ref	This represents the environment which is required to use the specified Processor.

Table B.27: BuildAction

Class	BuildActionEnvironment			
Note	This meta-class represents the ability to specify a build action environment.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BuildActionManifest.buildActionEnvironment			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
sdg	Sdg	*	aggr	This represents a general data structure intended to denote parameters for the BuildActionEnvironment.

Table B.28: BuildActionEnvironment

Class	**BuildActionManifest**			
Note	This meta-class represents the ability to specify a manifest for processing artifacts. An example use case is the processing of ECUC parameter values. **Tags:** atp.recommendedPackage=BuildActionManifests xml.globalElement=false			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
buildAction	BuildAction	*	aggr	This represents a particular action in the build chain. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=buildAction.shortName, buildAction.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
buildAction Environment	BuildActionEnvironment	*	aggr	This represents a build action environment. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=buildActionEnvironment.shortName, buildActionEnvironment.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
dynamicAction	BuildAction	*	ref	This denotes an Action which is to be executed as part of the dynamic action set.
startAction	BuildAction	*	ref	This specifies the list of actions to be performed at the beginning of the process. **Tags:** xml.sequenceOffset=-90
tearDownAction	BuildAction	*	ref	This specifies the set of action which shall be performed after all other actions in the manifest were performed. **Tags:** xml.sequenceOffset=-80

Table B.29: BuildActionManifest

Class	**ClientServerInterface**			
Note	A client/server interface declares a number of operations that can be invoked on a server by a client. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operation	ClientServerOperation	*	aggr	`ClientServerOperation`(s) of this `ClientServerInterface`. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=operation.shortName, operation.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime This Attribute is only used by the AUTOSAR Classic Platform.
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface.

Table B.30: ClientServerInterface

Class	**ClientServerInterfaceToBswModuleEntryBlueprintMapping**			
Note	This represents a mapping between one ClientServerInterface blueprint and BswModuleEntry blueprint in order to express the intended implementation of ClientServerOperations by specific BswModuleEntries under consideration of PortDefinedArguments. Such a mapping enables the formal check whether the number of arguments and the data types of arguments of the operation + additional PortDefined Arguments matches the signature of the BswModuleEntry. **Tags:** atp.recommendedPackage=BlueprintMappingSets This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Interface	ClientServerInterface	1	ref	The referenced ClientServerInterface represents the client server interface the mapping is dedicated to.
operation Mapping	ClientServerOperation BlueprintMapping	1..*	aggr	This specifies the operations used in the mapping between the ClientServerInterface and the BswModule Entry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=operationMapping, operation Mapping.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portDefined Argument Blueprint (ordered)	PortDefinedArgument Blueprint	*	aggr	This specifies the PortDefinedArguments used in the mapping between the ClientServerInterface and the Bsw ModuleEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portDefinedArgumentBlueprint, portDefined ArgumentBlueprint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.31: ClientServerInterfaceToBswModuleEntryBlueprintMapping

Class	**ClientServerOperation**			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.command, *AtpClassifier*.atpFeature, ClientServerInterface.operation, Diagnostic DataElementInterface.read, DiagnosticDataIdentifierInterface.read, DiagnosticDataIdentifierInterface. write, DiagnosticExtendedDataRecordInterface.provide, DiagnosticRoutineInterface.requestResult, DiagnosticRoutineInterface.start, DiagnosticRoutineInterface.stop, PhmRecoveryActionInterface. recovery, ServiceInterface.method			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this `ClientServerOperation`. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=argument.shortName, argument.variation Point.shortLabel 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 `ClientServerOperation` does not have to consider the usage of a shared buffer. This Attribute is only used by the AUTOSAR Classic Platform.
possibleError	ApplicationError	*	ref	Possible errors that may by raised by the referring operation. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.32: ClientServerOperation

Class	**Collection**			
Note	This meta-class specifies a collection of elements. A collection can be utilized to express additional aspects for a set of elements. Note that Collection is an ARElement. Therefore it is applicable e.g. for EvaluatedVariant, even if this is not obvious. Usually the category of a Collection is "SET". On the other hand, a Collection can also express an arbitrary relationship between elements. This is denoted by the category "RELATION" (see also [TPS_GST_00347]). In this case the collection represents an association from "sourceElement" to "targetElement" in the role "role". **Tags:** atp.recommendedPackage=Collections			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
autoCollect	AutoCollectEnum	0..1	attr	This attribute reflects how far the referenced objects are part of the collection. **Tags:** xml.sequenceOffset=20
collected Instance	AtpFeature	*	iref	This instance ref supports the use case that a particular instance is part of the collection. **Tags:** xml.sequenceOffset=60 **InstanceRef implemented by:** AnyInstanceRef
collection Semantics	NameToken	0..1	attr	Provides the ability to express the semantics of a Collection depending on the intended use case. The collectionSemantics is specified as a NameToken which must be agreed by all stakeholders. **Tags:** xml.sequenceOffset=25
element	Identifiable	*	ref	This is an element in the collection. Note that Collection itself is collectable. Therefore collections can be nested. In case of category="RELATION" this represents the target end of the relation. **Tags:** xml.sequenceOffset=40

▽

△

Class	Collection			
elementRole	Identifier	0..1	attr	This attribute allows to denote a particular role of the collection. Note that the applicable semantics shall be mutually agreed between the two parties. In particular it denotes the role of element in the context of sourceElement. **Tags:** xml.sequenceOffset=30
sourceElement	Identifiable	*	ref	Only if Category = "RELATION". This represents the source of a relation. **Tags:** xml.sequenceOffset=50
sourceInstance	AtpFeature	*	iref	Only if Category = "RELATION". This represents the source instance of a relation. **Tags:** xml.sequenceOffset=70 **InstanceRef implemented by:** AnyInstanceRef

Table B.33: Collection

Class	CompuMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:** atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=compuInternalToPhys xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=compuPhysToInternal xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:** xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:** xml.sequenceOffset=30

Table B.34: CompuMethod

Class	CompuScale			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	*ARObject*			
Aggregated by	CompuScales.compuScale			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**CompuScale**			
a2lDisplayText	String	0..1	attr	The value of this attribute shall be taken for generating one display text (specifically the OutVal) within the equivalent of the enclosing `CompuMethod` in A2L.
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. **Tags:** xml.sequenceOffset=60
compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:** xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveUnlimitedInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:** xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:** xml.sequenceOffset=20
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. **Tags:** xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table B.35: CompuScale

Class	**ConsistencyNeeds**			
Note	This meta-class represents the ability to define requirements on the implicit communication behavior.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ConsistencyNeedsBlueprintSet.consistencyNeeds, *SwComponentType*.consistencyNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ConsistencyNeeds**			
dpgDoesNot Require Coherency	DataPrototypeGroup	*	aggr	This group of VariableDataPrototypes does not require coherency with respect to the implicit communication behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dpgDoesNotRequireCoherency.shortName, dpgDoesNotRequireCoherency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dpgRequires Coherency	DataPrototypeGroup	*	aggr	This group of VariableDataPrototypes requires coherency with respect to the implicit communication behavior, i.e. all read and write access to VariableDataPrototypes in the DataPrototypeGroup by the RunnableEntitys of the RunnableEntityGroup need to be handled in a coherent manner. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dpgRequiresCoherency.shortName, dpgRequiresCoherency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
regDoesNot RequireStability	RunnableEntityGroup	*	aggr	This group of RunnableEntities does not require stability with respect to the implicit communication behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=regDoesNotRequireStability.shortName, regDoesNotRequireStability.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
regRequires Stability	RunnableEntityGroup	*	aggr	This group of RunnableEntities requires stability with respect to the implicit communication behavior, i.e. all read and write access to VariableDataPrototypes in the DataPrototypeGroup by the RunnableEntitys of the RunnableEntityGroup need to be handled in a stable manner. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=regRequiresStability.shortName, regRequiresStability.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.36: ConsistencyNeeds

Class	**ConsistencyNeedsBlueprintSet**			
Note	This meta class represents the ability to specify a set of blueprint for ConsistencyNeeds. **Tags:** atp.recommendedPackage=ConsistencyNeedsBlueprintSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
consistency Needs	ConsistencyNeeds	*	aggr	This represents a particular blueprint of consistency Needs. Note that it is **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=consistencyNeeds.shortName, consistencyNeeds.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.37: ConsistencyNeedsBlueprintSet

Class	DataConstr			
Note	This meta-class represents the ability to specify constraints on data. **Tags:** atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table B.38: DataConstr

Class	DataPrototypeGroup			
Note	This meta-class represents the ability to define a collection of DataPrototypes that are subject to the formal definition of implicit communication behavior. The definition of the collection can be nested.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ConsistencyNeeds.dpgDoesNotRequireCoherency, ConsistencyNeeds.dpgRequiresCoherency			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataPrototypeGroup	DataPrototypeGroup	*	iref	This represents the ability to define nested groups of VariableDataPrototypes. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataPrototypeGroup.contextSwComponentPrototype, dataPrototypeGroup.targetDataPrototypeGroup, dataPrototypeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime **InstanceRef implemented by:** InnerDataPrototypeGroupInCompositionInstanceRef
implicitDataAccess	VariableDataPrototype	*	iref	This represents a collection of VariableDataPrototypes that belong to the enclosing DataPrototypeGroup **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implicitDataAccess.contextSwComponentPrototype, implicitDataAccess.contextPortPrototype, implicitDataAccess.targetVariableDataPrototype, implicitDataAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime **InstanceRef implemented by:** VariableDataPrototypeInCompositionInstanceRef

Table B.39: DataPrototypeGroup

Class	DataTypeMappingSet
Note	This class represents a list of mappings between `ApplicationDataType`s and `ImplementationDataType`s. In addition, it can contain mappings between `ImplementationDataType`s and `ModeDeclarationGroup`s. **Tags:** atp.recommendedPackage=DataTypeMappingSets
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**DataTypeMappingSet**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an `ApplicationDataType` and its `AbstractImplementationDataType`.
modeRequest TypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an `ModeDeclarationGroup` and its `AbstractImplementationDataType`.

Table B.40: DataTypeMappingSet

Class	**Documentation**			
Note	This meta-class represents the ability to handle a so called standalone documentation. Standalone means, that such a documentation is not embedded in another AUTOSARElement or identifiable object. The standalone documentation is an entity of its own which denotes its context by reference to other objects and instances. **Tags:** atp.recommendedPackage=Documentations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context	DocumentationContext	*	aggr	This is the context of the particular documentation.
documentation Content	PredefinedChapter	0..1	aggr	This is the content of the documentation related to the specified contexts. **Tags:** xml.sequenceOffset=200

Table B.41: Documentation

Class	«atpMixed» **DocumentationBlock**			
Note	This class represents a documentation block. It is made of basic text structure elements which can be displayed in a table cell.			
Base	*ARObject*			
Aggregated by	ApplicabilityInfo.remark, AUTOSAR.introduction, BlueprintGenerator.introduction, *BlueprintPolicy* *Modifiable*.blueprintDerivationGuide, ClientServerOperationBlueprintMapping.blueprintMappingGuide, *DataMapping*.introduction, DefItem.def, *Describable*.introduction, EcucAddInfoParamValue.value, Ecu ResourceEstimation.introduction, Entry.entryContents, FrameMapping.introduction, *GeneralAnnotation*. annotationText, *Identifiable*.introduction, IPduMapping.introduction, ISignalMapping.introduction, Item. itemContents, LabeledItem.itemContents, LifeCycleInfo.remark, *MappingConstraint*.introduction, Msr QueryP2.msrQueryResultP2, Note.noteText, PortDefinedArgumentBlueprint.blueprintMappingGuide, PrmChar.cond, PrmChar.remark, *ScheduleTableEntry*.introduction, *SignalPathConstraint*.introduction, StructuredReq.conflicts, StructuredReq.dependencies, StructuredReq.description, StructuredReq. rationale, StructuredReq.remark, StructuredReq.supportingMaterial, StructuredReq.useCase, SwAxis Type.swGenericAxisDesc, TopicContent.blockLevelContent, TraceableText.text, VariationPoint.blueprint Condition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defList	DefList	0..1	aggr	This represents a definition list in the documentation block. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=40

▽

△

Class	«atpMixed» **DocumentationBlock**			
figure	MIFigure	0..1	aggr	This represents a figure in the documentation block. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=70
formula	MIFormula	0..1	aggr	This is a formula in the definition block. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=60
labeledList	LabeledList	0..1	aggr	This represents a labeled list. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=50
list	List	0..1	aggr	This represents numbered or unnumbered list. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=30
msrQueryP2	MsrQueryP2	0..1	aggr	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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=80
p	MultiLanguage Paragraph	0..1	aggr	This is one particular paragraph. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=10
structuredReq	StructuredReq	0..1	aggr	This aggregation supports structured requirements embedded in a documentation block. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=100
trace	TraceableText	0..1	aggr	This represents traceable text in the documentation block. This allows to specify requirements/constraints in any documentation block. The kind of the trace is specified in the category. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=90
verbatim	MultiLanguageVerbatim	0..1	aggr	This represents one particular verbatim text. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=20

Table B.42: DocumentationBlock

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

Table B.43: EcucAbstractStringParamDef

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

Table B.44: EcucBooleanParamDef

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

Table B.45: EcucChoiceReferenceDef

Class	***EcucContainerDef*** (abstract)
Note	Base class used to gather common attributes of configuration container definitions. This Class is only used by the AUTOSAR Classic Platform.
Base	*ARObject*, *AtpDefinition*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	EcucChoiceContainerDef, EcucParamConfContainerDef

▽

△

Class	***EcucContainerDef*** (abstract)			
Aggregated by	EcucDestinationUriPolicy.container, EcucModuleDef.container, EcucParamConfContainerDef.sub Container			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationUri	EcucDestinationUriDef	*	ref	Several destinationUris can be defined for an Ecuc ContainerDef. With such destinationUris an Ecuc ContainerDef is applicable for several EcucUriReference Defs. **Stereotypes:** atpUriDef
multiplicity ConfigClass	EcucMultiplicity ConfigurationClass	*	aggr	Specifies which MultiplicityConfigurationClass this container is available for which ConfigurationVariant. This aggregation is optional if the surrounding EcucModuleDef has the Category STANDARDIZED_MODULE_ DEFINITION. If the category attribute of the EcucModule Def is set to VENDOR_SPECIFIC_MODULE_ DEFINITION and if the upperMultiplicity is greater than the lowerMultiplicity then this aggregation is mandatory. **Tags:** xml.name Plural=MULTIPLICITY-CONFIG-CLASSES
origin	String	0..1	attr	This attribute specifies whether this configuration container is an AUTOSAR standardized container or whether it is vendor-specific.
postBuildVariant Multiplicity	Boolean	0..1	attr	Indicates if a container may have different number of instances in different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
requiresIndex	Boolean	0..1	attr	Used to define whether the value element for this definition shall be provided with an index.

Table B.46: EcucContainerDef

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

▽

△

Class	EcucContainerValue			
subContainer	EcucContainerValue	*	aggr	Aggregates all sub-containers within this container. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=subContainer.shortName, sub Container.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table B.47: EcucContainerValue

Class	EcucDefinitionCollection			
Note	This represents the anchor point of an ECU Configuration Parameter Definition within the AUTOSAR templates structure. **Tags:** atp.recommendedPackage=EcucDefs This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
module	EcucModuleDef	*	ref	References to the module definitions of individual software modules.

Table B.48: EcucDefinitionCollection

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

Table B.49: EcucEnumerationParamDef

Class	EcucFloatParamDef			
Note	Configuration parameter type for Float. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucDestinationUriPolicy.parameter, EcucParamConfContainerDef.parameter			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**EcucFloatParamDef**			
defaultValue	Float	0..1	attr	Default value of the float configuration parameter. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime
max	Limit	0..1	attr	Max value allowed for the parameter defined. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime
min	Limit	0..1	attr	Min value allowed for the parameter defined. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime

Table B.50: EcucFloatParamDef

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

Table B.51: EcucIntegerParamDef

Class	**EcucModuleDef**			
Note	Used as the top-level element for configuration definition for Software Modules, including BSW and RTE as well as ECU Infrastructure. **Tags:** atp.recommendedPackage=EcucDefs This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpDefinition*, *CollectableElement*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
apiServicePrefix	CIdentifier	0..1	attr	For modules where several instances of the VSMD can be defined the apiServicePrefix defines the API namespace of the derived instances, e.g. Cdd, Xfrm (ComXf, SomeIpXf, E2EXf).
container	EcucContainerDef	*	aggr	Aggregates the top-level container definitions of this specific module definition. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=container.shortName xml.sequenceOffset=11
postBuildVariant Support	Boolean	0..1	attr	Indicates if a module supports different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.

▽

△

Class	**EcucModuleDef**			
refinedModule Def	EcucModuleDef	0..1	ref	Optional reference from the Vendor Specific Module Definition to the Standardized Module Definition it refines. In case this EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION this reference shall not be provided. In case this EcucModuleDef has the category VENDOR_SPECIFIC_MODULE_ DEFINITION this reference is mandatory. **Stereotypes:** atpUriDef
supported ConfigVariant	EcucConfiguration VariantEnum	*	attr	Specifies which ConfigurationVariants are supported by this software module. This attribute is optional if the Ecuc ModuleDef has the category STANDARDIZED_ MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_ MODULE_DEFINITION then this attribute is mandatory.

Table B.52: EcucModuleDef

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

Table B.53: EcucNumericalParamValue

Class	***EcucParameterDef*** (abstract)			
Note	Abstract class used to define the similarities of all ECU Configuration Parameter types defined as subclasses. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*EcucAbstractStringParamDef*, EcucAddInfoParamDef, EcucBooleanParamDef, EcucEnumerationParamDef, EcucFloatParamDef, EcucIntegerParamDef			
Aggregated by	EcucDestinationUriPolicy.parameter, EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
derivation	EcucDerivation Specification	0..1	aggr	A derivation of a Configuration Parameter value can be specified by an informal Calculation Formula or by a formal language that can be used to specify the computational rules.
symbolicName Value	Boolean	0..1	attr	Specifies that this parameter's value is used, together with the aggregating container, to derive a symbolic name definition. See chapter "Representation of Symbolic Names" in Ecuc specification for more details.

▽

△

Class	***EcucParameterDef*** (abstract)			
withAuto	Boolean	0..1	attr	Specifies whether it shall be allowed on the value side to specify this parameter value as "AUTO". If withAuto is "true" it shall be possible to set the "isAuto Value" attribute of the respective parameter to "true". This means that the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards. These implicit updated values might require a re-generation of other modules which reference these values. If withAuto is "false" it shall not be possible to set the "is AutoValue" attribute of the respective parameter to "true". If withAuto is not present the default is "false".

Table B.54: EcucParameterDef

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

Table B.55: EcucParameterValue

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

Table B.56: EcucReferenceDef

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

Table B.57: EcucTextualParamValue

Class	EcucUriReferenceDef			
Note	Definition of reference with a destination that is specified via a destinationUri. With such a reference it is possible to define a reference to a EcucContainerDef in a different module independent from the concrete definition of the target container. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractInternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucDestinationUriPolicy.reference, EcucParamConfContainerDef.reference			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destinationUri	EcucDestinationUriDef	0..1	ref	Any EcucContainerDef with a destinationUri that is identical to the destinationUri that is referenced here defines a valid target. **Stereotypes:** atpUriDef

Table B.58: EcucUriReferenceDef

Class	FlatMap			
Note	Contains a flat list of references to software objects. This list is used to identify instances and to resolve name conflicts. The scope is given by the RootSwCompositionPrototype for which it is used, i.e. it can be applied to a system, system extract or ECU-extract. An instance of FlatMap may also be used in a preliminary context, e.g. in the scope of a software component before integration into a system. In this case it is not referred by a RootSwCompositionPrototype. **Tags:** atp.recommendedPackage=FlatMaps This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
instance	FlatInstanceDescriptor	*	aggr	A descriptor instance aggregated in the flat map. The variation point accounts for the fact, that the system in scope can be subject to variability, and thus the existence of some instances is variable. The aggregation has been made splitable because the content might be contributed by different stakeholders at different times in the workflow. Plus, the overall size might be so big that eventually it becomes more manageable if it is distributed over several files. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=instance.shortName, instance.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table B.59: FlatMap

Class	***Identifiable*** (abstract)			
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, AppOsTask ProxyToEcuTaskProxyMapping, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartition Mapping, AppliedStandard, AsynchronousServerCallResultPoint, *AtpBlueprint*, *AtpBlueprintable*, *Atp Classifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BinaryManifest AddressableObject*, BinaryManifestItemDefinition, *BinaryManifestResource*, BinaryManifestResource Definition, BlockState, BswInternalTriggeringPoint, BswModuleDependency, *BuildActionEntity*, Build ActionEnvironment, CanTpAddress, CanTpChannel, CanTpNode, Chapter, ClientIdDefinition, Client ServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEvent Group, *CouplingElementAbstractDetails*, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPort StructuralElement*, *CpSoftwareClusterResource*, CpSoftwareClusterResourceToApplicationPartition Mapping, CpSoftwareClusterToApplicationPartitionMapping, CpSoftwareClusterToEcuInstanceMapping, CpSoftwareClusterToResourceMapping, *CryptoServiceMapping*, CyclicHandlingComDataToOsTask ProxyMapping, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, DataTransformation, *Dds AbstractServiceInstanceElementCp*, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, Diagnostic ExtendedDataRecordElement, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DltApplication, DltArgument, DltArgumentProps, DltLogChannel, Dlt Message, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, ECUMapping, *EOCExecutableEntity RefAbstract*, EcuPartition, EcuPartitionToCoreMapping, EcucContainerValue, *EcucDefinitionElement*, EcucDestinationUriDef, EcucEnumerationLiteralDef, EcucQuery, EcucValidationCondition, Ethernet WakeupSleepOnDatalineConfig, EventHandler, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeature Relation, FMFeatureRestriction, FMFeatureSelection, FlatInstanceDescriptor, FlexrayArTpNode, Flexray TpConnectionControl, FlexrayTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, Global TimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HeapUsage*, HwAttributeDef, HwAttributeLiteral Def, HwPin, HwPinGroup, *IEEE1722TpAcfBus*, *IEEE1722TpAcfBusPart*, IPSecRule, IPv6ExtHeader FilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, ImpositionTime, InternalTriggering Point, J1939Node, J1939SharedAddressCluster, J1939TpNode, Keyword, LifeCycleState, LinSchedule Table, LinTpNode, Linker, MacAddressVlanMembership, MacMulticastGroup, MacSecKayParticipant, Mc DataInstance, MemorySection, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, Mode SwitchSenderComSpecProps, NetworkEndpoint, *NmCluster*, NmEcu, *NmNode*, NvBlockDescriptor, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, Pdu Triggering, PerInstanceMemory, *PhysicalChannel*, PortElementToCommunicationResourceMapping, PortGroup, *PortInterfaceMapping*, QueuedReceiverComSpecProps, ResourceConsumption, RootSw CompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RteEventInCompositionSeparation, RteEventIn CompositionToOsTaskProxyMapping, RteEventInSystemSeparation, RteEventInSystemToOsTaskProxy Mapping, RunnableEntityGroup, *SdgAttribute*, SdgClass, SecOcJobRequirement, SecureCommunication AuthenticationProps, SecureCommunicationFreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, *ServerCallPoint*, ServerComSpecProps, *ServiceNeeds*, SignalService TranslationElementProps, SignalServiceTranslationEventProps, SignalServiceTranslationProps, Socket Address, SomeipTpChannel, *StackUsage*, StaticSocketConnection, StructuredReq, SwGenericAxis ParamType, SwServiceArg, SwcServiceDependency, SwcToApplicationPartitionMapping, SwcToEcu Mapping, SwcToImplMapping, SwitchAsynchronousTrafficShaperGroupEntry, SwitchAtsInstanceEntry, SwitchFlowMeteringEntry, SwitchStreamFilterActionDestPortModification, SwitchStreamFilterEntry, SwitchStreamFilterRule, SwitchStreamGateEntry, SwitchStreamIdentification, SystemMapping, System SignalGroupToCommunicationResourceMapping, SystemSignalToCommunicationResourceMapping, TDCpSoftwareClusterMapping, TDCpSoftwareClusterResourceMapping, TcpOptionFilterList, *Timing Clock*, TimingClockSyncAccuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, Timing ExtensionResource, TimingModeInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, TransformationISignalPropsIdent, *TransformationProps*, TransformationTechnology, Trigger, VariableAccess, VariationPointProxy, View Map, VlanConfig, WaitPoint			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	***Identifiable*** (abstract)			
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table B.60: Identifiable

Class	**ImplementationDataType**
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**ImplementationDataType**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to true, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of `ImplementationDataTypeElement` is subject to variability with the purpose to support the conditional existence of elements inside a `ImplementationDataType` representing a structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=subElement.shortName, subElement.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Stereotypes:** atpSplitable **Tags:** 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 B.61: ImplementationDataType

Class	**ImpositionTime**			
Note	This meta class represents one particular imposition time.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ImpositionTimeDefinitionGroup.impositionTime			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.62: ImpositionTime

Class	**Keyword**			
Note	This meta-class represents the ability to predefine keywords which may subsequently be used to construct names following a given naming convention, e.g. the AUTOSAR naming conventions. Note that such names is not only shortName. It could be symbol, or even longName. Application of keywords is not limited to particular names.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	KeywordSet.keyword			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
abbrName	NameToken	1	attr	This attribute specifies an abbreviated name of a keyword. This abbreviation may e.g. be used for constructing valid shortNames according to the AUTOSAR naming conventions. Unlike shortName, it may contain any name token. E.g. it may consist of digits only.

▽

△

Class	Keyword			
classification	NameToken	*	attr	This attribute allows to attach classification to the Keyword such as MEAN, ACTION, CONDITION, INDEX, PREPOSITION

Table B.63: Keyword

Class	KeywordSet			
Note	This meta--class represents the ability to collect a set of predefined keywords. **Tags:** atp.recommendedPackage=KeywordSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
keyword	Keyword	*	aggr	This is one particular keyword in the keyword set.

Table B.64: KeywordSet

Class	LifeCycleInfo			
Note	LifeCycleInfo describes the life cycle state of an element together with additional information like what to use instead			
Base	*ARObject*			
Aggregated by	LifeCycleInfoSet.lifeCycleInfo			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
lcObject	Referrable	1	ref	Element(s) have the life cycle as described in lcState.
lcState	LifeCycleState	0..1	ref	This denotes the particular state assigned to the object. If no lcState is given then the default life cycle state of Life CycleInfoSet is assumed.
periodBegin	LifeCyclePeriod	0..1	aggr	Starting point of period in which the element has the denoted life cycle state lcState. If no periodBegin is given then the default period begin of LifeCycleInfoSet is assumed.
periodEnd	LifeCyclePeriod	0..1	aggr	Expiry date, i.e. end point of period the element does not have the denoted life cycle state lcState any more. If no periodEnd is given then the default period begin of Life CycleInfoSet is assumed.
remark	DocumentationBlock	0..1	aggr	Remark describing for example • why the element was given the specified life cycle • the semantics of useInstead
useInstead	Referrable	*	ref	Element(s) that should be used instead of the one denoted in referrable. Only relevant in case of life cycle states lcState unlike "valid". In case there are multiple references the exact semantics shall be individually described in the remark.

Table B.65: LifeCycleInfo

Class	LifeCycleInfoSet
Note	This meta class represents the ability to attach a life cycle information to a particular set of elements. The information can be defined for a particular period. This supports the definition of transition plans. If no period is specified, the life cycle state applies forever. **Tags:** atp.recommendedPackage=LifeCycleInfoSets

▽

△

Class	LifeCycleInfoSet			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
defaultLcState	LifeCycleState	1	ref	This denotes the default life cycle state. To be used in all LifeCycleInfo elements within the LifeCycleInfoSet if no life cycle state is stated there explicitly. I.e. the defaultLc State can be overwritten in LifeCycleInfo elements.
defaultPeriod Begin	LifeCyclePeriod	0..1	aggr	Default starting point of period in which all the specified lifeCycleInfo apply. Note that the default period can be overridden for each lifeCycleInfo individually.
defaultPeriod End	LifeCyclePeriod	0..1	aggr	Default expiry date, i.e. default end point of period for which all specified lifeCycleInfo apply. Note that the default period can be overridden for each lifeCycleInfo individually.
lifeCycleInfo	LifeCycleInfo	*	aggr	This represents one particular life cycle information.
usedLifeCycle StateDefinition Group	LifeCycleStateDefinition Group	1	ref	This denotes the life cycle states applicable to the current life cycle info set.

Table B.66: LifeCycleInfoSet

Class	LifeCycleState			
Note	This meta class represents one particular state in the LifeCycle.			
Base	ARObject, AtpBlueprint, AtpBlueprintable, Identifiable, MultilanguageReferrable, Referrable			
Aggregated by	LifeCycleStateDefinitionGroup.lcState			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.67: LifeCycleState

Class	LifeCycleStateDefinitionGroup			
Note	This meta class represents the ability to define the states and properties of one particular life cycle. **Tags:** atp.recommendedPackage=LifeCycleStateDefinitionGroups			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
lcState	LifeCycleState	*	aggr	Describes a single life cycle state of this life cycle state definition group.

Table B.68: LifeCycleStateDefinitionGroup

Class	ModeDeclarationGroup
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.recommendedPackage=ModeDeclarationGroups
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDesignElement, UploadablePackageElement
Aggregated by	ARPackage.element

▽

△

Class	**ModeDeclarationGroup**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialMode	ModeDeclaration	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeDeclaration.shortName, mode Declaration.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
modeManager ErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user). This Attribute is only used by the AUTOSAR Classic Platform.
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup This Attribute is only used by the AUTOSAR Classic Platform.
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). This Attribute is only used by the AUTOSAR Classic Platform.
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. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.69: ModeDeclarationGroup

Class	***MultilanguageReferrable*** (abstract)			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders). They also may have a longName. But they are not considered to contribute substantially to the overall structure of an AUTOSAR description. In particular it does not contain other Referrables.			
Base	*ARObject*, *Referrable*			
Subclasses	Caption, DefItem, DocumentationContext, *Identifiable*, SdgCaption, *TraceReferrable*, *Traceable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
longName	MultilanguageLong Name	0..1	aggr	This specifies the long name of the object. Long name is targeted to human readers and acts like a headline.

Table B.70: MultilanguageReferrable

Class	**NonqueuedReceiverComSpec**			
Note	Communication attributes specific to non-queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**NonqueuedReceiverComSpec**			
aliveTimeout	TimeValue	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. This Attribute is only used by the AUTOSAR Classic Platform.
enableUpdate	Boolean	0..1	attr	This attribute controls whether application code is entitled to check whether the value of the corresponding Variable DataPrototype has been updated. This Attribute is only used by the AUTOSAR Classic Platform.
filter	DataFilter	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. This Attribute is only used by the AUTOSAR Classic Platform.
handleNever Received	Boolean	0..1	attr	This attribute specifies whether for the corresponding VariableDataPrototype the "never received" flag is available. If yes, the RTE is supposed to assume that initially the VariableDataPrototype has not been received before. After the first reception of the corresponding VariableDataPrototype the flag is cleared. • If the value of this attribute is set to "true" the flag is required. • If set to "false", the RTE shall not support the "never received" functionality for the corresponding Variable DataPrototype. This Attribute is only used by the AUTOSAR Classic Platform.
handleTimeout Type	HandleTimeoutEnum	0..1	attr	This attribute controls the behavior with respect to the handling of timeouts. This Attribute is only used by the AUTOSAR Classic Platform.
initValue	ValueSpecification	0..1	aggr	Initial value to be used in case the sending component is not yet initialized. If the sender also specifies an initial value, then the receiver's value will be used. This Attribute is only used by the AUTOSAR Classic Platform.
returnNoNew DataEnabled	Boolean	0..1	attr	This attribute defines whether the RTE API functions related to the RPortPrototype shall return No New Data Error if no new data is received from COM. This Attribute is only used by the AUTOSAR Classic Platform.
timeout Substitution Value	ValueSpecification	0..1	aggr	This attribute represents the substitution value applicable in the case of a timeout. This Attribute is only used by the AUTOSAR Classic Platform.
transportError CountEnabled	Boolean	0..1	attr	This attribute defines whether the RTE API functions related to the RPortPrototype shall return the number of transport errors (i.e. COM, SecOC errors) that happened since the last call of the respective API. This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	NonqueuedReceiverComSpec			
valueErrorCount Enabled	Boolean	0..1	attr	This attribute defines whether the RTE API functions related to the RPortPrototype shall return the number of value errors (i.e. out of range, invalid value) that happened since the last call of the respective API. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.71: NonqueuedReceiverComSpec

Class	NonqueuedSenderComSpec			
Note	Communication attributes for non-queued sender/receiver communication (sender side)			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataFilter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
initValue	ValueSpecification	0..1	aggr	Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.

Table B.72: NonqueuedSenderComSpec

Class	NvBlockSwComponentType			
Note	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. **Tags:** atp.recommendedPackage=SwComponentTypes This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bulkNvData Descriptor	BulkNvDataDescriptor	*	aggr	This aggregation formally defines the bulk Nv Blocks that are provided to the application software by the enclosing NvBlockSwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bulkNvDataDescriptor.shortName, bulkNvDataDescriptor.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
nvBlock Descriptor	NvBlockDescriptor	*	aggr	Specification of the properties of exactly one NVRAM Block. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=nvBlockDescriptor.shortName, nvBlockDescriptor.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.73: NvBlockSwComponentType

Class	***PPortComSpec*** (abstract)			
Note	Communication attributes of a provided `PortPrototype`. This class will contain attributes that are valid for all kinds of provide ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	ModeSwitchSenderComSpec, NvProvideComSpec, ParameterProvideComSpec, *SenderComSpec*, ServerComSpec			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.74: PPortComSpec

Class	**PPortPrototype**			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provided Interface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType

Table B.75: PPortPrototype

Class	**PRPortPrototype**			
Note	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *Atp Feature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provided Required Interface	PortInterface	0..1	tref	This represents the `PortInterface` used to type the `PRPortPrototype`. **Stereotypes:** isOfType

Table B.76: PRPortPrototype

Class	***PackageableElement*** (abstract)			
Note	This meta-class specifies the ability to be a member of an AUTOSAR package.			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ARElement*, EnumerationMappingTable, *FibexElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.77: PackageableElement

Class	**PortDefinedArgumentValue**
Note	A PortDefinedArgumentValue is passed to a RunnableEntity dealing with the ClientServerOperations provided by a given PortPrototype. Note that this is restricted to PPortPrototypes of a ClientServer Interface.
Base	*ARObject*

▽

△

Class	PortDefinedArgumentValue			
Aggregated by	PortAPIOption.portArgValue			
Attribute	Type	Mult.	Kind	Note
value	ValueSpecification	0..1	aggr	Specifies the actual value.
valueType	ImplementationDataType	0..1	tref	The implementation type of this argument value. It should not be composite type or a pointer. **Stereotypes:** isOfType

Table B.78: PortDefinedArgumentValue

Class	PortInterface (abstract)			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Subclasses	ClientServerInterface, DataInterface, ModeSwitchInterface, TriggerInterface			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
isService	Boolean	0..1	attr	This flag is set if the `PortInterface` is to be used for communication between an • `ApplicationSwComponentType` or • `ServiceProxySwComponentType` or • `SensorActuatorSwComponentType` or • `ComplexDeviceDriverSwComponentType` • `ServiceSwComponentType` • `EcuAbstractionSwComponentType` and a `ServiceSwComponentType` (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime This Attribute is only used by the AUTOSAR Classic Platform.
serviceKind	ServiceProviderEnum	0..1	attr	This attribute provides further details about the nature of the applied service. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.79: PortInterface

Class	PortInterfaceMapping (abstract)			
Note	Specifies one `PortInterfaceMapping` to support the connection of Ports typed by two different `PortInterface`s with PortInterface elements having unequal names and/or unequal semantic (resolution or range).			
Base	ARObject, AtpBlueprint, AtpBlueprintable, Identifiable, MultilanguageReferrable, Referrable			
Subclasses	ClientServerInterfaceMapping, ModeInterfaceMapping, TriggerInterfaceMapping, VariableAndParameterInterfaceMapping			
Aggregated by	PortInterfaceMappingSet.portInterfaceMapping			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table B.80: PortInterfaceMapping

Class	**PortInterfaceMappingSet**			
Note	Specifies a set of (one or more) `PortInterfaceMapping`s. **Tags:** atp.recommendedPackage=PortInterfaceMappingSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
portInterface Mapping	PortInterfaceMapping	*	aggr	Specifies one `PortInterfaceMapping` to support the connection of `PortPrototype`s typed by two different `PortInterface`s with `PortInterface` elements having unequal names and/or unequal semantic (resolution or range). **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portInterfaceMapping.shortName, portInterfaceMapping.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime

Table B.81: PortInterfaceMappingSet

Class	***PortPrototype*** (abstract)			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this `PortPrototype` with respect to client/server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=senderReceiverAnnotation
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table B.82: PortPrototype

Class	***RPortComSpec*** (abstract)
Note	Communication attributes of a required PortPrototype. This class will contain attributes that are valid for all kinds of require-ports, independent of client-server or sender-receiver communication patterns.
Base	*ARObject*
Subclasses	ClientComSpec, ModeSwitchReceiverComSpec, NvRequireComSpec, ParameterRequireComSpec, *ReceiverComSpec*

▽

△

Class	***RPortComSpec*** (abstract)			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.83: RPortComSpec

Class	**RPortPrototype**			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
mayBe Unconnected	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. This Attribute is only used by the AUTOSAR Classic Platform.
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table B.84: RPortPrototype

Class	***Referrable*** (abstract)			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, TpConnectionIdent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table B.85: Referrable

Class	**RunnableEntity**
Note	A `RunnableEntity` represents the smallest code-fragment that is provided by an `AtomicSwComponentType` and are executed under control of the RTE. `RunnableEntity`s are for instance set up to respond to data reception or operation invocation on a server.
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**RunnableEntity**			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.runnable			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=asynchronousServerCallResultPoint.shortName, asynchronousServerCallResultPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime This Attribute is only used by the AUTOSAR Classic Platform.
canBeInvoked 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 `AtomicSwComponentType`). This implies that it is the responsibility of the implementation of the `RunnableEntity` to take care of this form of concurrency.
dataRead Access	VariableAccess	*	aggr	RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReadAccess.shortName, dataReadAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointBy Argument	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature. The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePointByArgument.shortName, dataReceivePointByArgument.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
dataReceive PointByValue	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of the return value. The aggregation of dataReceivePointBy Value is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePointByValue.shortName, data ReceivePointByValue.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataSendPoint.shortName, dataSend Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataWrite Access	VariableAccess	*	aggr	RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataWriteAccess.shortName, dataWrite Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
external TriggeringPoint	ExternalTriggeringPoint	*	aggr	The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=externalTriggeringPoint.ident.shortName, externalTriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
internal TriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPoint.shortName, internal TriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
modeAccess Point	ModeAccessPoint	*	aggr	The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeAccessPoint.ident.shortName, mode AccessPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeSwitch Point	ModeSwitchPoint	*	aggr	The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeSwitchPoint.shortName, modeSwitch Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
parameter Access	ParameterAccess	*	aggr	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. 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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterAccess.shortName, parameter Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
readLocal Variable	VariableAccess	*	aggr	The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of read LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=readLocalVariable.shortName, readLocal Variable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
serverCallPoint	ServerCallPoint	*	aggr	The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serverCallPoint.shortName, serverCall Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	**RunnableEntity**			
symbol	CIdentifier	0..1	attr	The symbol describing this `RunnableEntity`'s entry point. This is considered the API of the `RunnableEntity` and is required during the RTE contract phase.
waitPoint	WaitPoint	*	aggr	The `WaitPoint` associated with the `RunnableEntity`.
writtenLocal Variable	VariableAccess	*	aggr	The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of written LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=writtenLocalVariable.shortName, written LocalVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.86: RunnableEntity

Class	**RunnableEntityGroup**			
Note	This meta-class represents the ability to define a collection of RunnableEntities. The collection can be nested.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ConsistencyNeeds.regDoesNotRequireStability, ConsistencyNeeds.reg RequiresStability			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
runnableEntity	RunnableEntity	*	iref	This represents a collection of RunnableEntitys that belong to the enclosing RunnableEntityGroup. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=runnableEntity.contextSwComponent Prototype, runnableEntity.targetRunnableEntity, runnable Entity.variationPoint.shortLabel vh.latestBindingTime=preCompileTime **InstanceRef implemented by:** RunnableEntityIn CompositionInstanceRef
runnableEntity Group	RunnableEntityGroup	*	iref	This represents the ability to define nested groups of RunnableEntitys. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=runnableEntityGroup.contextSwComponent Prototype, runnableEntityGroup.targetRunnableEntity Group, runnableEntityGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime **InstanceRef implemented by:** InnerRunnableEntity GroupInCompositionInstanceRef

Table B.87: RunnableEntityGroup

Class	**SdgClass**
Note	An SdgClass specifies the name and structure of the SDG that may be used to store proprietary data in an AUTOSAR model. The SdgClass is similar to an UML stereotype.

▽

△

Class	**SdgClass**			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgElementWithGid*			
Aggregated by	SdgDef.sdgClass			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
attribute (ordered)	SdgAttribute	*	aggr	Defintion of the structure of the Sdg **Tags:** xml.sequenceOffset=30
caption	Boolean	0..1	attr	Specifies if a caption is required. Note: only Sdgs that have a caption can be referenced **Tags:** xml.sequenceOffset=20
extendsMeta Class	MetaClassName	0..1	attr	The AUTOSAR Meta-Class that may be extended by this SdgClass. **Tags:** xml.sequenceOffset=10
sdgConstraint	TraceableText	*	ref	Semantic constraints that restrict the structure of the special data group. **Tags:** xml.sequenceOffset=40

Table B.88: SdgClass

Class	**SdgDef**			
Note	A SdgDef groups several SdgClasses which belong to the same extension. The concept of an SdgDef is similiar to an UML Profile. **Tags:** atp.recommendedPackage=SdgDefs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sdgClass	SdgClass	*	aggr	The owned sdgClasses which define the structure of the Sdgs **Tags:** xml.namePlural=SDG-CLASSES

Table B.89: SdgDef

Primitive	**SectionInitializationPolicyType**
Note	SectionInitializationPolicyType describes the intended initialization of MemorySections. The following values are standardized in AUTOSAR Methodology: • **INIT**: To be used for (explicitly or not explicitly) initialized variables. • **CLEARED**: To be used for 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. Please note that the values are defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility. **Tags:** xml.xsd.customType=SECTION-INITIALIZATION-POLICY-TYPE xml.xsd.type=NMTOKEN

Table B.90: SectionInitializationPolicyType

Class	**SenderReceiverInterface**			
Note	A sender/receiver interface declares a number of data elements to be sent and received. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	*	aggr	The data elements of this `SenderReceiverInterface`.
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement
metaDataItem Set	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with `dataElement`s of the enclosing `SenderReceiverInterface`

Table B.91: SenderReceiverInterface

Enumeration	**StandardNameEnum**
Note	This enumeration lists all allowed standard abbreviations.
Aggregated by	AppliedStandard.appliesTo, StructuredReq.appliesTo
Literal	***Description***
AP	This values represents the Adaptive Platform. **Tags:** atp.EnumerationLiteralIndex=0
CP	This Value represents the Classic Platform. **Tags:** atp.EnumerationLiteralIndex=1
FO	This values represents the Foundation. **Tags:** atp.EnumerationLiteralIndex=2

Table B.92: StandardNameEnum

Class	**StructuredReq**			
Note	This represents a structured requirement. This is intended for a case where specific requirements for features are collected. Note that this can be rendered as a labeled list.			
Base	*ARObject*, *DocumentViewSelectable*, *Identifiable*, *MultilanguageReferrable*, *Paginateable*, *Referrable*, *Traceable*			
Aggregated by	DocumentationBlock.structuredReq			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
appliesTo	StandardNameEnum	*	attr	This attribute represents the platform the requirement is assigned to. **Tags:** xml.namePlural=APPLIES-TO-DEPENDENCIES xml.sequenceOffset=25
conflicts	DocumentationBlock	0..1	aggr	This represents an informal specification of conflicts. **Tags:** xml.sequenceOffset=40
date	DateTime	1	attr	This represents the date when the requirement was initiated. **Tags:** xml.sequenceOffset=5
dependencies	DocumentationBlock	0..1	aggr	This represents an informal specification of dependencies. Note that upstream tracing should be formalized in the property trace provided by the superclass Traceable. **Tags:** xml.sequenceOffset=30

△

Class	**StructuredReq**			
description	DocumentationBlock	0..1	aggr	This represents the general description of the requirement. **Tags:** xml.sequenceOffset=10
importance	String	1	attr	This allows to represent the importance of the requirement. **Tags:** xml.sequenceOffset=8
issuedBy	String	1	attr	This represents the person, organization or authority which issued the requirement. **Tags:** xml.sequenceOffset=6
rationale	DocumentationBlock	0..1	aggr	This represents the rationale of the requirement. **Tags:** xml.sequenceOffset=20
remark	DocumentationBlock	0..1	aggr	This represents an informal remark. Note that this is not modeled as annotation, since these remark is still essential part of the requirement. **Tags:** xml.sequenceOffset=60
supporting Material	DocumentationBlock	0..1	aggr	This represents an informal specification of the supporting material. **Tags:** xml.sequenceOffset=50
testedItem	Traceable	*	ref	This association represents the ability to trace on the same specification level. This supports for example the of acceptance tests. **Tags:** xml.sequenceOffset=70
type	String	1	attr	This attribute allows to denote the type of requirement to denote for example is it an "enhancement", "new feature" etc. **Tags:** xml.sequenceOffset=7
useCase	DocumentationBlock	0..1	aggr	This describes the relevant use cases. Note that formal references to use cases should be done in the trace relation. **Tags:** xml.sequenceOffset=35

Table B.93: StructuredReq

Class	**SwAddrMethod**			
Note	Used to assign a common addressing method, e.g. common memory section, to data or code objects. These objects could actually live in different modules or components. **Tags:** atp.recommendedPackage=SwAddrMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memory Allocation KeywordPolicy	MemoryAllocation KeywordPolicyType	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 MemorySection in with the related objects shall be placed. These properties are handled as to be selected. The intended options are mentioned in the list. In the Memory Mapping configuration, this option list is used to determine an appropriate MemMapAddressing ModeSet.

▽

△

Class	**SwAddrMethod**			
section Initialization Policy	SectionInitialization PolicyType	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	MemorySectionType	0..1	attr	Defines the type of memory sections which can be associated with this addressing method.

Table B.94: SwAddrMethod

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

Table B.95: SwBaseType

Class	**SwComponentPrototype**			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.component			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType

Table B.96: SwComponentPrototype

Class	***SwComponentType*** (abstract)			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***SwComponentType*** (abstract)			
consistency Needs	ConsistencyNeeds	*	aggr	This represents the collection of `ConsistencyNeeds` owned by the enclosing `SwComponentType`. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=consistencyNeeds.shortName, consistency Needs.variationPoint.shortLabel vh.latestBindingTime=preCompileTime This Attribute is only used by the AUTOSAR Classic Platform.
port	PortPrototype	*	aggr	The `PortPrototype`s through which this `SwComponentType` can communicate. The aggregation of `PortPrototype` is subject to variability with the purpose to support the conditional existence of `PortPrototype`s. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=port.shortName, port.variationPoint.short Label vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portGroup.shortName, portGroup.variation Point.shortLabel vh.latestBindingTime=preCompileTime
swcMapping Constraint	SwComponentMapping Constraints	*	ref	Reference to constraints that are valid for this Sw ComponentType. This Attribute is only used by the AUTOSAR Classic Platform.
swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the `SwComponentType`. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10
unitGroup	UnitGroup	*	ref	This allows for the specification of which `UnitGroup`s are relevant in the context of referencing `SwComponentType`. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.97: SwComponentType

Class	**SwServiceArg**			
Note	Specifies the properties of a data object exchanged during the call of an SwService, e.g. an argument or a return value. The SwServiceArg can also be used in the argument list of a C-macro. For this purpose the category shall be set to "MACRO". A reference to implementationDataType can optional be added if the actual argument has an implementationDataType.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BswModuleEntry.argument, BswModuleEntry.returnType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SwServiceArg**			
direction	ArgumentDirection Enum	0..1	attr	Specifies the direction of the data transfer. The direction shall indicate the direction of the actual information that is being consumed by the caller and/or the callee, not the direction of formal arguments in C. The attribute is optional for backwards compatibility reasons. For example, if a pointer is used to pass a memory address for the expected result, the direction shall be "out". If a pointer is used to pass a memory address with content to be read by the callee, its direction shall be "in". **Tags:** xml.sequenceOffset=10
swArraysize	ValueList	0..1	aggr	This turns the argument of the service to an array. **Tags:** xml.sequenceOffset=20
swDataDef Props	SwDataDefProps	0..1	aggr	Data properties of this SwServiceArg. **Tags:** xml.sequenceOffset=30

Table B.98: SwServiceArg

Class	**SwcBswMapping**			
Note	Maps an SwcInternalBehavior to an BswInternalBehavior. This is required to coordinate the API generation and the scheduling for AUTOSAR Service Components, ECU Abstraction Components and Complex Driver Components by the RTE and the BSW scheduling mechanisms. **Tags:** atp.recommendedPackage=SwcBswMappings This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswBehavior	BswInternalBehavior	0..1	ref	The mapped BswInternalBehavior
runnable Mapping	SwcBswRunnable Mapping	*	aggr	A mapping between a pair of SWC and BSW runnables. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=runnableMapping, runnable Mapping.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
swcBehavior	SwcInternalBehavior	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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=synchronizedModeGroup, synchronized ModeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
synchronized Trigger	SwcBswSynchronized Trigger	*	aggr	A pair of SWC and BSW Triggers to be synchronized by the scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=synchronizedTrigger, synchronized Trigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.99: SwcBswMapping

Class	SwcInternalBehavior			
Note	The SwcInternalBehavior of an AtomicSwComponentType describes the relevant aspects of the software-component with respect to the RTE, i.e. the RunnableEntitys and the RTEEvents they respond to.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *InternalBehavior*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtomicSwComponentType*.internalBehavior, *AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arTypedPerInstanceMemory	VariableDataPrototype	*	aggr	Defines an AUTOSAR typed memory-block that needs to be available for each instance of the SW-component. This is typically only useful if supportsMultipleInstantiation is set to "true" or if the component defines NVRAM access via permanent blocks. The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the software component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arTypedPerInstanceMemory.shortName, arTypedPerInstanceMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
event	RTEEvent	*	aggr	This is a RTEEvent specified for the particular SwcInternalBehavior. The aggregation of RTEEvent is subject to variability with the purpose to support the conditional existence of RTEEvents. Note: the number of RTEEvents might vary due to the conditional existence of PortPrototypes using DataReceivedEvents or due to different scheduling needs of algorithms. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=event.shortName, event.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
exclusiveAreaPolicy	SwcExclusiveAreaPolicy	*	aggr	Options how to generate the ExclusiveArea related APIs. When no SwcExclusiveAreaPolicy is specified for an ExclusiveArea the default values apply. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaPolicy, exclusiveAreaPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
explicitInterRunnableVariable	VariableDataPrototype	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of explicitInterRunnableVariable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=explicitInterRunnableVariable.shortName, explicitInterRunnableVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

△

Class	SwcInternalBehavior			
implicitInter Runnable Variable	VariableDataPrototype	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of implicitInterRunnable Variable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implicitInterRunnableVariable.shortName, implicitInterRunnableVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedData TypeSet	IncludedDataTypeSet	*	aggr	The `includedDataTypeSet` is used by a software component for its implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedDataTypeSet
includedMode Declaration GroupSet	IncludedMode DeclarationGroupSet	*	aggr	This aggregation represents the included Mode DeclarationGroups **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedModeDeclarationGroupSet
instantiation DataDefProps	InstantiationDataDef Props	*	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". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=instantiationDataDefProps, instantiationData DefProps.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstance Memory	PerInstanceMemory	*	aggr	Defines a per-instance memory object needed by this software component. The aggregation of PerInstance Memory is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceMemory.shortName, perInstance Memory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstance Parameter	ParameterData Prototype	*	aggr	Defines parameter(s) or characteristic value(s) that needs to be available for each instance of the software-component. This is typically only useful if supportsMultipleInstantiation is set to "true". The aggregation of perInstanceParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceParameter.shortName, per InstanceParameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**SwcInternalBehavior**			
portAPIOption	PortAPIOption	*	aggr	Options for generating the signature of port-related calls from a runnable to the RTE and vice versa. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portAPIOption.port, portAPIOption.variation Point.shortLabel vh.latestBindingTime=preCompileTime
runnable	RunnableEntity	*	aggr	This is a `RunnableEntity` specified for the particular `SwcInternalBehavior`. The aggregation of `RunnableEntity` is subject to variability with the purpose to support the conditional existence of `RunnableEntity`s. Note: the number of `RunnableEntity`s might vary due to the conditional existence of `PortPrototype`s using `DataReceivedEvent`s or due to different scheduling needs of algorithms. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=runnable.shortName, runnable.variation Point.shortLabel vh.latestBindingTime=preCompileTime
service Dependency	SwcService Dependency	*	aggr	Defines the requirements on AUTOSAR Services for a particular item. The aggregation of SwcServiceDependency is subject to variability with the purpose to support the conditional existence of ports as well as the conditional existence of ServiceNeeds. The SwcServiceDependency owned by an 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 <<atp Splitable>>. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceDependency.shortName, service Dependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
shared Parameter	ParameterData Prototype	*	aggr	Defines parameter(s) or characteristic value(s) shared between SwComponentPrototypes of the same Sw ComponentType The aggregation of sharedParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sharedParameter.shortName, shared Parameter.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	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=variationPointProxy.shortName

Table B.100: SwcInternalBehavior

Class	***TDEventVfbPort*** (abstract)			
Note	A `TimingDescriptionEvent` occuring on a `PortPrototype`.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventModeDeclaration, TDEventOperation, TDEventTrigger, TDEventVariableDataPrototype			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isExternal	Boolean	0..1	attr	This attribute is used to refer to external events that are related to hardware I/O, like physical sensors and actuators, at Virtual Functional Bus (VFB) level. This Attribute is only used by the AUTOSAR Classic Platform.
portPrototype	PortPrototype	0..1	iref	`PortPrototype` on which the TimingEvent occurs **Tags:** atp.Status=draft **InstanceRef implemented by:** PortInCompositionType InstanceRef
portPrototype Blueprint	PortPrototypeBlueprint	0..1	ref	port on which the TimingEvent shall apply (in the context of an AUTOSAR blueprint)

Table B.101: TDEventVfbPort

Class	***Traceable*** (abstract)			
Note	This meta class represents the ability to be subject to tracing within an AUTOSAR model. Note that it is expected that its subclasses inherit either from MultilanguageReferrable or from Identifiable. Nevertheless it also inherits from MultilanguageReferrable in order to provide a common reference target for all Traceables.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Subclasses	StructuredReq, *TimingConstraint*, TraceableTable, TraceableText			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
trace	Traceable	*	ref	This association represents the ability to trace to upstream requirements / constraints. This supports for example the bottom up tracing ProjectObjectives <- MainRequirements <- Features <- RequirementSpecs <- BSW/AI **Tags:** xml.sequenceOffset=20

Table B.102: Traceable

Class	**TraceableTable**			
Note	This meta-class represents a table which can be referenced in order to establish requirements tracing. It supports specific kinds of tracing such as requirements / constraints. The following approach applies: • `shortName`: represents the tag for tracing • `longName`: represents the headline • `category`: represents the kind of the tagged table			
Base	*ARObject*, *DocumentViewSelectable*, *Identifiable*, *MultilanguageReferrable*, *Paginateable*, *Referrable*, *Traceable*			
Aggregated by	TopicContent.traceableTable			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
table	Table	1	aggr	This represents a table with a traceable table.

Table B.103: TraceableTable

Class	TraceableText			
Note	Represents a paragraph level text which can be referenced in order to establish tracing. It supports specific tracing of document items as specified in [TPS_STDT_00098]. The following approach applies: • `shortName`: represents the tag for tracing • `longName`: represents the headline • `category`: represents the kind of the tagged text			
Base	*ARObject*, *DocumentViewSelectable*, *Identifiable*, *MultilanguageReferrable*, *Paginateable*, *Referrable*, *Traceable*			
Aggregated by	DocumentationBlock.trace			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
text	DocumentationBlock	1	aggr	This represents the text to which the tag applies. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table B.104: TraceableText

Class	VariableDataPrototype			
Note	A `VariableDataPrototype` represents a formalized generic piece of information that is typically mutable by the application software layer. `VariableDataPrototype` is used in various contexts and the specific context gives the otherwise generic `VariableDataPrototype` a dedicated semantics.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.indication, *AtpClassifier*.atpFeature, BswInternalBehavior.arTypedPerInstanceMemory, BswModuleDescription.providedData, BswModuleDescription.requiredData, BulkNvDataDescriptor.bulkNvBlock, DiagnosticSovdAccessArgument.contentObject, *InternalBehavior*.staticMemory, NvBlockDescriptor.ramBlock, NvDataInterface.nvData, SenderReceiverInterface.dataElement, ServiceInterface.event, SwcInternalBehavior.arTypedPerInstanceMemory, SwcInternalBehavior.explicitInterRunnableVariable, SwcInternalBehavior.implicitInterRunnableVariable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table B.105: VariableDataPrototype

Class	VfbTiming			
Note	A model element used to define timing descriptions and constraints at VFB level. TimingDescriptions aggregated by VfbTiming are restricted to event chains referring to events which are derived from the class TDEventVfb. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *TimingExtension*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
component	SwComponentType	0..1	ref	This defines the scope of a VfbTiming. All corresponding timing descriptions and constraints shall be defined within this scope.

Table B.106: VfbTiming

C Variation Points in this Template

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 specific 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 processed in a tool, please refer to [2]

Variation Point	*Latest Binding Time*
BlueprintPolicyList.maxNumberOfElements	blueprintDerivationTime
BlueprintPolicyList.minNumberOfElements	blueprintDerivationTime
ClientServerInterfaceToBswModuleEntryBlueprintMapping.operationMapping	preCompileTime
ClientServerInterfaceToBswModuleEntryBlueprintMapping.portDefinedArgument Blueprint	preCompileTime
ConsistencyNeedsBlueprintSet.consistencyNeeds	preCompileTime
SwDataDefProps	codeGenerationTime
SwDataDefProps.swValueBlockSize	preCompileTime
SwDataDefProps.swValueBlockSizeMult	preCompileTime
SwTextProps.swMaxTextSize	preCompileTime
ValueList.vf	preCompileTime

Table C.1: Usage of variation points

D Change History

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

This chapter provides the change history of traceable items in this document. The lists also include traceable items that have been removed from the document in a later version. These items do not appear as hyperlinks in the document.

D.1 Change History of this document according to AUTOSAR Release R4.3.1

D.1.1 Added Specification Items in 4.3.1

none

D.1.2 Changed Specification Items in 4.3.1

none

D.1.3 Deleted Specification Items in 4.3.1

none

D.1.4 Added Constraints in 4.3.1

none

D.1.5 Changed Constraints in 4.3.1

none

D.1.6 Deleted Constraints in 4.3.1

none

D.2 Change History of this document according to AUTOSAR Release R4.4.0

D.2.1 Added Specification Items in 4.4.0

Number	Heading
[TPS_STDT_00092]	Return values of the `BlueprintFormula`.ecuc query
[TPS_STDT_00211]	Specification of the AUTOSAR Standards that are part of the `Baseline`

Table D.1: Added Specification Items in 4.4.0

D.2.2 Changed Specification Items in 4.4.0

Number	Heading
[TPS_STDT_00006]	Applying Expression Pattern
[TPS_STDT_00021]	Specialization of `BlueprintFormula`
[TPS_STDT_00045]	Transferring Objects in General
[TPS_STDT_00047]	Ignore Blueprint Attributes in Non Blueprints
[TPS_STDT_00086]	Specify a name pattern or a blueprint value in blueprints

Table D.2: Changed Specification Items in 4.4.0

D.2.3 Deleted Specification Items in 4.4.0

none

D.2.4 Added Constraints in 4.4.0

Number	Heading
[constr_2625]	Allowed uptraces wrt. life cycles

Table D.3: Added Constraints in 4.4.0

D.2.5 Changed Constraints in 4.4.0

Number	Heading
[constr_2553]	`shortName` shall follow the pattern defined in the Blueprint
[constr_2554]	Derived objects shall match the blueprints
[constr_2569]	Purely Blueprint Motivated `VariationPoint`s

Table D.4: Changed Constraints in 4.4.0

D.2.6 Deleted Constraints in 4.4.0

none

D.3 Change History of this document according to AUTOSAR Release R19-11

D.3.1 Added Specification Items in 19-11

none

D.3.2 Changed Specification Items in 19-11

Number	Heading
[TPS_STDT_00006]	Applying Expression Pattern
[TPS_STDT_00021]	Specialization of `BlueprintFormula`
[TPS_STDT_00028]	Resolving `VariationPoint` in Blueprints
[TPS_STDT_00030]	Blueprint of `VariationPoint`
[TPS_STDT_00044]	Transferring `VariationPoint`
[TPS_STDT_00046]	Configuration dependent properties
[TPS_STDT_00048]	Express Decisions when Deriving Objects

Table D.5: Changed Specification Items in 19-11

D.3.3 Deleted Specification Items in 19-11

none

D.3.4 Added Constraints in 19-11

none

D.3.5 Changed Constraints in 19-11

Number	Heading
[constr_2556]	No Blueprint Motivated `VariationPoint`s in AUTOSAR Descriptions
[constr_2569]	Purely Blueprint Motivated `VariationPoint`s

Table D.6: Changed Constraints in 19-11

D.3.6 Deleted Constraints in 19-11

none

D.4 Change History of this document according to AUTOSAR Release R20-11

D.4.1 Added Specification Items in R20-11

none

D.4.2 Changed Specification Items in R20-11

none

D.4.3 Deleted Specification Items in R20-11

none

D.4.4 Added Constraints in R20-11

none

D.4.5 Changed Constraints in R20-11

Number	Heading
[constr_2540]	Tagged text category

Table D.7: Changed Constraints in R20-11

D.4.6 Deleted Constraints in R20-11

none

D.5 Change History of this document according to AUTOSAR Release R21-11

D.5.1 Added Specification Items in R21-11

none

D.5.2 Changed Specification Items in R21-11

Number	Heading
[TPS_STDT_00014]	Blueprinting `BswModuleEntry`
[TPS_STDT_00027]	Blueprinting `BswModuleDescription`
[TPS_STDT_00064]	Applied Life Cycle Information Sets on AUTOSAR provided Models (M1)
[TPS_STDT_00081]	Representation of constraint items in AUTOSAR template documents
[TPS_STDT_00090]	Blueprinting `BswEntryRelationshipSet`
[TPS_STDT_00091]	Blueprinting `BswEntryRelationshipSet`
[TPS_STDT_00092]	Return values of the `BlueprintFormula`.ecuc query
[TPS_STDT_00107]	Validation Semantics of global `ConcreteClassTailoring`. `multiplicityRestriction` with `validationRoot`==true
[TPS_STDT_00115]	Analysis of Tool Compatibility
[TPS_STDT_00157]	Purpose of `DataFormatTailoring`

Table D.8: Changed Specification Items in R21-11

D.5.3 Deleted Specification Items in R21-11

none

D.5.4 Added Constraints in R21-11

none

D.5.5 Changed Constraints in R21-11

none

D.5.6 Deleted Constraints in R21-11

none

D.6 Change History of this document according to AUTOSAR Release R22-11

D.6.1 Added Specification Items in R22-11

Number	Heading
[TPS_STDT_00093]	Representation of advisory items in AUTOSAR template documents
[TPS_STDT_00094]	Sentence pattern

Table D.9: Added Specification Items in R22-11

D.6.2 Changed Specification Items in R22-11

Number	Heading
[TPS_STDT_00042]	namePattern for `shortName`s of `TraceableText` in Standardization Documents
[TPS_STDT_00050]	namePattern for AUTOSAR delivered Files
[TPS_STDT_00078]	Representation of requirements in AUTOSAR documents

Table D.10: Changed Specification Items in R22-11

D.6.3 Deleted Specification Items in R22-11

none

D.6.4 Added Constraints in R22-11

none

D.6.5 Changed Constraints in R22-11

none

D.6.6 Deleted Constraints in R22-11

none

D.7 Change History of this document according to AUTOSAR Release R23-11

D.7.1 Added Specification Items in R23-11

Number	Heading
[TPS_STDT_00095]	Semantics of an `ImpositionTime`
[TPS_STDT_00096]	Application of an `ImpositionTime`
[TPS_STDT_00097]	Semantics of an unspecified `ImpositionTime`

Table D.11: Added Specification Items in R23-11

D.7.2 Changed Specification Items in R23-11

Number	Heading
[TPS_STDT_00021]	Specialization of `BlueprintFormula`
[TPS_STDT_00042]	namePattern for `shortName`s of `TraceableText` in Standardization Documents
[TPS_STDT_00081]	Representation of constraint items in AUTOSAR template documents
[TPS_STDT_00089]	Identifying specification items which are constraints in AUTOSAR ASWS/SWS/PRS documents
[TPS_STDT_00092]	Return values of the `BlueprintFormula.ecuc` query
[TPS_STDT_00198]	Default `multiplicityRestriction` of `Meta-Attribute`s (when not explicitly specified)
[TPS_STDT_00203]	Default `PrimitiveAttributeTailoring.valueRestriction` (when not explicitly specified)

Table D.12: Changed Specification Items in R23-11

D.7.3 Deleted Specification Items in R23-11

Number	Heading
[TPS_STDT_00111]	AUTOSAR Standardized Constraints

Table D.13: Deleted Specification Items in R23-11

D.7.4 Added Constraints in R23-11

none

D.7.5 Changed Constraints in R23-11

Number	Heading
[constr_2540]	Tagged text category

Table D.14: Changed Constraints in R23-11

D.7.6 Deleted Constraints in R23-11

Number	Heading
[constr_2564]	`VariationPoint` in Blueprints of `PackageableElement`

Table D.15: Deleted Constraints in R23-11

D.8 Change History of this document according to AUTOSAR Release R24-11

D.8.1 Added Specification Items in R24-11

Number	Heading
[TPS_STDT_00098]	Standardized `category`s of `TraceableText` and `TraceableTable`

Table D.16: Added Specification Items in R24-11

D.8.2 Changed Specification Items in R24-11

Number	Heading
[TPS_STDT_00042]	namePattern for `shortName`s of `TraceableText` in Standardization Documents
[TPS_STDT_00050]	Standardized naming convention for released AUTOSAR files
[TPS_STDT_00078]	Representation of requirements in AUTOSAR documents

Table D.17: Changed Specification Items in R24-11

D.8.3 Deleted Specification Items in R24-11

Number	Heading
[TPS_STDT_00029]	Representation of test items in AUTOSAR documents
[TPS_STDT_00059]	TraceableText

Table D.18: Deleted Specification Items in R24-11

D.8.4 Added Constraints in R24-11

none

D.8.5 Changed Constraints in R24-11

none

D.8.6 Deleted Constraints in R24-11

Number	Heading
[constr_2540]	Tagged text category

Table D.19: Deleted Constraints in R24-11

D.9 Change History of this document according to AUTOSAR Release R25-11

D.9.1 Added Specification Items in R25-11

Number	Heading
[TPS_STDT_00099]	Standardized naming convention for trace groups
[TPS_STDT_00137]	AUTOSAR Document Names and Abbreviations for Trace Prefixes
[TPS_STDT_00148]	SPECIFICATION_ITEMs which are not-applicable for up-tracing
[TPS_STDT_00149]	AUTOSAR reserved shortNames under the top-level ARPackage=AUTOSAR
[TPS_STDT_00150]	AUTOSAR document categories
[TPS_STDT_00151]	AUTOSAR document meta-data

Table D.20: Added Specification Items in R25-11

D.9.2 Changed Specification Items in R25-11

Number	Heading
[TPS_STDT_00042]	Naming convention for SPECIFICATION_ITEMs
[TPS_STDT_00050]	Standardized naming convention for published AUTOSAR files
[TPS_STDT_00088]	Representation of constraint items in AUTOSAR non template documents
[TPS_STDT_00089]	SPECIFICATION_ITEMs with constraint semantics

Table D.21: Changed Specification Items in R25-11

D.9.3 Deleted Specification Items in R25-11

Number	Heading
[TPS_STDT_00057]	Representation of generally fulfilled requirements in AUTOSAR documents
[TPS_STDT_00058]	Representation of under specified requirements in AUTOSAR documents
[TPS_STDT_00100]	Motivation of Description of Data Exchange Points
[TPS_STDT_00101]	Compatibility of ConcreteClassTailorings
[TPS_STDT_00102]	Referencing AUTOSAR Specification Elements via shortName
[TPS_STDT_00103]	Referencing AUTOSAR Specification Elements via alternativeName
[TPS_STDT_00104]	Referencing Custom Specification Elements
[TPS_STDT_00105]	Serialized Profile
[TPS_STDT_00106]	Effective Profile
[TPS_STDT_00107]	Validation Semantics of global ConcreteClassTailoring.multiplicityRestriction with validationRoot==true
[TPS_STDT_00108]	Validation Semantics of global ConcreteClassTailoring.multiplicityRestriction with validationRoot==false
[TPS_STDT_00109]	AUTOSAR Standardized Concrete Meta-Classes
[TPS_STDT_00110]	Identification of Potential Interoperability Issues
[TPS_STDT_00112]	Validation Semantics of ClassTailoring.multiplicityRestriction in the context of AggregationTailoring.typeTailoring
[TPS_STDT_00113]	Validation Semantics of AbstractClassTailoring.multiplicityRestriction
[TPS_STDT_00114]	MultiplicityRestrictionWithSeverity in the context of ClassTailoring vs. AggregationTailoring/ReferenceTailoring
[TPS_STDT_00115]	Analysis of Tool Compatibility
[TPS_STDT_00116]	Limitation of Analysis of Profile of Data Exchange Points
[TPS_STDT_00117]	Agreed Profile of Data Exchange Point
[TPS_STDT_00118]	Compliance with Profile of Data Exchange Point
[TPS_STDT_00119]	Validation Semantics of ClassTailoring.multiplicityRestriction in the context of ReferenceTailoring.typeTailoring
[TPS_STDT_00120]	Purpose of DataExchangePoint

▽

△

Number	Heading
[TPS_STDT_00121]	High-level Overview Description of `DataExchangePoint`
[TPS_STDT_00122]	Purpose of `Baseline`
[TPS_STDT_00123]	Guidance on how to specify `SpecificationDocumentScope` and `DocumentElementScope`
[TPS_STDT_00124]	Purpose of `SpecElementScope`
[TPS_STDT_00125]	Trigger for Evaluation of Constraints
[TPS_STDT_00126]	Definition: Data Format Elements
[TPS_STDT_00127]	Validation Environment
[TPS_STDT_00128]	Compatibility of `SpecificationDocumentScope`s
[TPS_STDT_00129]	Semantics of `DataFormatElementScope` with `inScope`==true
[TPS_STDT_00130]	Navigation strategy for validation
[TPS_STDT_00131]	Compatibility of `AggregationTailoring`
[TPS_STDT_00132]	Purpose of `SdgTailoring`
[TPS_STDT_00133]	Compatibility of `ReferenceTailoring`
[TPS_STDT_00134]	Compatibility of `PrimitiveAttributeTailoring`
[TPS_STDT_00135]	Compatibility of `ClassContentConditional`
[TPS_STDT_00136]	Compatibility of `AttributeTailoring`
[TPS_STDT_00138]	Purpose of `ReferenceTailoring`
[TPS_STDT_00139]	AUTOSAR Standardized References of Meta-Class
[TPS_STDT_00140]	Purpose of `AggregationTailoring`
[TPS_STDT_00141]	AUTOSAR Standardized Aggregations of Meta-Class
[TPS_STDT_00142]	Purpose of `PrimitiveAttributeTailoring`
[TPS_STDT_00143]	AUTOSAR Standardized Primitive Attributes of Meta-Class
[TPS_STDT_00144]	Purpose of `AttributeTailoring`
[TPS_STDT_00145]	Purpose of `ClassTailoring`
[TPS_STDT_00146]	AUTOSAR Standardized Abstract Meta-Classes
[TPS_STDT_00147]	Purpose of `ConstraintTailoring`
[TPS_STDT_00156]	Purpose of `SpecificationScope`
[TPS_STDT_00157]	Purpose of `DataFormatTailoring`
[TPS_STDT_00159]	Semantics of Attribute that is in Scope
[TPS_STDT_00160]	Compatibility of `DocumentElementScope`s
[TPS_STDT_00163]	Validation Semantics of `ConcreteClassTailoring`
[TPS_STDT_00164]	Semantics of a Constraint that is out of Scope
[TPS_STDT_00165]	Semantics of Constraint that is in Scope
[TPS_STDT_00167]	Semantics of SdgTailoring that is in scope
[TPS_STDT_00168]	Share documentation of Rationale
[TPS_STDT_00169]	Handling of unresolved references
[TPS_STDT_00170]	Local documentation of Rationale

▽

△

Number	Heading
[TPS_STDT_00172]	Purpose of `RestrictionWithSeverity`
[TPS_STDT_00173]	Purpose of `ValueRestrictionWithSeverity`
[TPS_STDT_00174]	Purpose of `MultiplicityRestrictionWithSeverity`
[TPS_STDT_00175]	Purpose of `VariationRestrictionWithSeverity`
[TPS_STDT_00176]	Context specific Tailoring
[TPS_STDT_00177]	Global `ClassTailoring`
[TPS_STDT_00178]	Role Specific `ClassTailoring`
[TPS_STDT_00179]	Conditional `ClassTailoring`
[TPS_STDT_00180]	Invariant Content Model
[TPS_STDT_00181]	Conditional Content Model
[TPS_STDT_00182]	Validation Semantics of `AbstractClassTailoring`
[TPS_STDT_00183]	Compatibility of `Baseline`s
[TPS_STDT_00186]	Scope and Restrictions of Data Format Elements
[TPS_STDT_00187]	Purpose of `DocumentElementScope`
[TPS_STDT_00188]	Purpose of `SpecificationDocumentScope`
[TPS_STDT_00190]	Default Scope of concrete `Meta Class`es
[TPS_STDT_00191]	Purpose of `Baseline Profile of Data Exchange Point`
[TPS_STDT_00192]	Default Scope of `AUTOSAR Specification`s
[TPS_STDT_00193]	Default Scope of `AUTOSAR Specification Elements`
[TPS_STDT_00195]	Default Scope of `Meta Attribute`s
[TPS_STDT_00196]	Default Validation Root of concrete `Meta Class`es
[TPS_STDT_00197]	Default `multiplicityRestriction` of `Meta-Class`es (when not explicitly specified)
[TPS_STDT_00198]	Default `multiplicityRestriction` of `Meta-Attribute`s (when not explicitly specified)
[TPS_STDT_00199]	Default `variationRestriction` of `Meta-Attribute`s (when not explicitly specified)
[TPS_STDT_00200]	Default `variationRestriction` of `Meta-Class`es with ≪`atpVariation`≫ (when not explicitly specified)
[TPS_STDT_00201]	Compatibility of `VariationRestrictionWithSeverity.variation`
[TPS_STDT_00202]	Compatibility of `VariationRestrictionWithSeverity.validBindingTime`
[TPS_STDT_00203]	Default `PrimitiveAttributeTailoring.valueRestriction` (when not explicitly specified)
[TPS_STDT_00204]	Default `PrimitiveAttributeTailoring.defaultValueHandling`
[TPS_STDT_00205]	Compatibility of `ValueRestrictionWithSeverity`
[TPS_STDT_00206]	Compatibility of `UnresolvedReferenceRestrictionWithSeverity`
[TPS_STDT_00207]	Default `ReferenceTailoring.unresolvedReferenceRestriction`
[TPS_STDT_00208]	Compatibility of `ConstraintTailoring`s
[TPS_STDT_00209]	Compatibility of `SdgTailoring`s

▽

△

Number	Heading
[TPS_STDT_00210]	Compatibility of `MultiplicityRestrictionWithSeverity`
[TPS_STDT_00211]	Specification of the AUTOSAR Standards that are part of the `Baseline`

Table D.22: Deleted Specification Items in R25-11

D.9.4 Added Constraints in R25-11

none

D.9.5 Changed Constraints in R25-11

none

D.9.6 Deleted Constraints in R25-11

Number	Heading
[constr_2608]	Custom extensions shall be part of the `Documentation` that is referenced by the `Baseline`
[constr_2609]	Single revision per AUTOSAR standard
[constr_2610]	No alternativeName if matching via `shortName`
[constr_2611]	Referenced AUTOSAR Specification Elements shall be part of the AUTOSAR Specification Baseline
[constr_2612]	`shortName` of `ConcreteClassTailoring` shall match the name of an AUTOSAR specified concrete meta-class
[constr_2613]	`shortName` of `AbstractClassTailoring` shall match the name of an AUTOSAR specified abstract meta-class
[constr_2614]	`PrimitiveAttributeCondition.attribute` shall reference invariant owned `PrimitiveAttributeTailoring`, only
[constr_2615]	`AggregationCondition.aggregation` shall reference invariant owned `AggregationTailoring`, only
[constr_2616]	`ReferenceCondition.reference` shall reference invariant owned `ReferenceTailoring`, only
[constr_2617]	`ClassTailoring.variationRestriction` only applicable for «atpVariation» classes
[constr_2618]	ShortName of AttributeTailoring shall match owned or inherited attributes
[constr_2619]	No `AttributeTailoring` for Derived or Abstract Attributes
[constr_2620]	`shortName` of `PrimitiveAttributeTailoring` shall be a primitive attribute in the referenced Baseline

▽

△

Number	Heading
[constr_2621]	The shortName of AggregationTailoring shall match the name of an AUTOSAR specified aggregation of the meta-class
[constr_2622]	The shortName of ReferenceTailoring shall match the name of an AUTOSAR specified reference of the meta-class
[constr_2623]	Referenced SdgClass shall be part of a SdgDef that is referenced by the Baseline
[constr_2624]	AttributeTailoring.variationRestriction only applicable for «atp Variation» attributes

Table D.23: Deleted Constraints in R25-11