Document Title	Specification of Manifest
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	713

Document Status	Final
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	19-03

Document Change History

Date	Release	Changed by	Description
2019-03-29	19-03	AUTOSAR Release Management	• Introduction of Diagnostic Port Interfaces • Overhaul of Software Cluster and introduction of Software Package • Support for Identity and Access Management • Network Management Configuration
2018-10-31	18-10	AUTOSAR Release Management	• Finish introduction of CppImplementationDataType • Support for optional elements in structures • Rework configuration of adaptive platform modules
2018-03-29	18-03	AUTOSAR Release Management	• Time Synchronization • DDS Deployment
2017-10-27	17-10	AUTOSAR Release Management	• Optional elements in Service Interfaces • Interaction with web services • Secure Communication • Support for interaction with crypto and persistency • Signal-to-Service translation • Support for E2E communication • Platform Health Management • Uploadable Software Package

2017-03-31	17-03	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

References

[1] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture

[3] Reference Model for Service Oriented Architecture 1.0
https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[4] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[5] Specification of RESTful communication
AUTOSAR_SWS_REST

[6] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[7] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes

[8] SOME/IP Protocol Specification
AUTOSAR_PRS_SOMEIPProtocol

[9] Specification of Persistency
AUTOSAR_SWS_Persistency

[10] IEEE Standard for Information Technology- Standardized Application Environment Profile (AEP)-POSIX Realtime and Embedded Application Support
https://standards.ieee.org/findstds/standard/1003.13-2003.html

[11] Specification of Time Synchronization for Adaptive Platform
AUTOSAR_SWS_TimeSync

[12] Specification of Communication Management
AUTOSAR_SWS_CommunicationManagement

[13] Explanation of ara::com API
AUTOSAR_EXP_ARAComAPI

[14] Specification of Platform Health Management for Adaptive Platform
AUTOSAR_SWS_PlatformHealthManagement

[15] Information technology – Universal Coded Character Set (UCS)
http://www.iso.org

[16] System Template
AUTOSAR_TPS_SystemTemplate

[17] Specification of Execution Management
AUTOSAR_SWS_ExecutionManagement

[18] Diagnostic Extract Template
AUTOSAR_TPS_DiagnosticExtractTemplate

[19] Specification of Diagnostics
AUTOSAR_SWS_Diagnostics

[20] Specification of ECU Resource Template
AUTOSAR_TPS_ECUResourceTemplate

[21] Road vehicles – Diagnostic communication over Internet Protocol (DoIP)
http://www.iso.org

[22] SOME/IP Service Discovery Protocol Specification
AUTOSAR_PRS_SOMEIPServiceDiscoveryProtocol

[23] Data Distribution Service (DDS), Version 1.4
http://www.omg.org/spec/DDS/1.4

[24] RPC over DDS, Version 1.0
https://www.omg.org/spec/DDS-RPC/1.0

[25] Specification of SW-C End-to-End Communication Protection Library
AUTOSAR_SWS_E2ELibrary

[26] Log and Trace Protocol Specification
AUTOSAR_PRS_LogAndTraceProtocol

[27] REST: Architectural Styles and the Design of Network-based Software Architectures

1 Introduction

This document contains the specification of the so-called the *Manifest* on the *AUTOSAR adaptive platform*. A description of the overall modeling approach can be found in section 1.1. A reference to the definition of the term *service* is given in section 1.2.

The term *Manifest* is used in this specification in the meaning of a formal specification of configuration content. Please find a more detailed description of the term and the implications for the *AUTOSAR adaptive platform* in section 2.

Please note that the content of the document (despite the name) extends to the description of design elements necessary to develop software for the *AUTOSAR adaptive platform*.

The design-related modeling mainly is focused on the development of application software on the *AUTOSAR adaptive platform* as well as the connection between application and diagnostics and is described in detail[1] in section 3 and section 4.

Section 5, in particular, describes the big picture of *AUTOSAR classic platform* and *AUTOSAR adaptive platform* communicating via service-oriented communication.

Section 6 describes the options for configuring a machine by means of a *manifest*.

Section 7 represents that counterpart to section 3 on deployment level, it describes the content of the so-called *execution manifest*.

Section 8 contains a string of sub-sections that explain the manifest content of platform module functionality.

Section 9 provides a detailed description of how service-oriented communication shall be configured on *manifest* level.

Section 10 explains how signal-based communication can be transformed into service-oriented communication and vice versa in order to participate in the communication between ECUs on the *AUTOSAR classic platform*.

Section 11 describes the modeling of communication with web services following the REST pattern

Section 12 describes the idea behind and the configuration of the concept of an uploadable software package.

[1]The description of the design elements may be moved to other model-related documents in the future. But for the time being, there is a coexistence of manifest-related and design-related model elements in this document.

1.1 Modeling Approach

The *AUTOSAR adaptive platform* has been introduced when the *AUTOSAR classic platform* was already a stable and well-established standard in the automotive domain.

And yet, the *AUTOSAR adaptive platform* is no successor of the *AUTOSAR classic platform*. Both platforms complement each other for specific use cases that can be better implemented by one or the other platform.

In this situation, two possible approaches for modeling on the *AUTOSAR adaptive platform* could have been taken:

- The *AUTOSAR adaptive platform* is based on different principles than the *AUTOSAR classic platform*, and hence the modeling approach could also **decouple from the canon of the AUTOSAR classic platform as much as possible** to advertise the fact that the two platforms have different purposes.

  Consequentially, even if specific model elements have clear counterparts in the respective other platform, use a different terminology to not confuse the users of both platforms.

- Despite the undeniable differences between the two platforms, there is still a significant number of striking similarities that strongly encourage the **usage of existing modeling concepts** from the *AUTOSAR classic platform*, especially from the specification of the AUTOSAR Software-Component Template [1], as much as possible.

  Consequentially, the conclusion is to use the identical meta-classes for similar purposes on both platforms. It will then be necessary to extend some of the affected meta-classes platform specific where applicable and add constraints that clarify the platform-specific usage of the mentioned extensions.

Without further ado, the modeling approach for the *AUTOSAR adaptive platform* follows the second alternative.

This means, for example, that a piece of application software on the *AUTOSAR adaptive platform* shall be represented by an `SwComponentType`. This includes the definition of `CompositionSwComponentType`s that in turn aggregate `SwComponentPrototype`s typed by e.g. (in case of the *AUTOSAR adaptive platform*) `AdaptiveApplicationSwComponentType`s.

This also means that an `AtomicSwComponentType` used on the *AUTOSAR adaptive platform* shall **not** aggregate `AtomicSwComponentType.internalBehavior` because the latter is reserved for usage on the *AUTOSAR classic platform*.

The reuse of existing model-elements for the definition of the meta-model for the *AUTOSAR adaptive platform* has the side effect that the descriptions of existing model elements may contain references to technical details that only make sense on the *AUTOSAR classic platform*.

After all, the model elements were created when only the *AUTOSAR classic platform* existed.

These references shall be taken with a grain of salt. It is expected that readers can abstract from those details and extract the aspects of these model elements that create relevance for the description of the *AUTOSAR adaptive platform*.

1.2 The Term Service

It is essential to keep in mind that the term *service* is frequently used within this document in particular and the *AUTOSAR adaptive platform* in general.

This usage has its reasons despite the fact that the meaning of the term *service* on the *AUTOSAR adaptive platform* collides with other meanings used within AUTOSAR.

In summary, the following meaning of the term *service* exist in the scope of AUTOSAR:

- The Term *service* is used in the layered software architecture [2] to denote the highest layer of the AUTOSAR software architecture that interacts with the application. In this context, model elements like `ServiceSwComponentType`, `SwcServiceDependency`, `ServiceNeeds`, or `PortInterface.isService` have been created on the *AUTOSAR classic platform*.
- The term *service* is used to express that information is related or required in a workshop where a car is **serviced**. In this context, *service-only diagnostic trouble codes* (DTC) are defined.
- The term *service* is used to describe the handling of **diagnostic services**, e.g. UDS service *ReadDataByIdentifier*, for the communication between a diagnostic tester and a diagnostic stack on an (AUTOSAR) ECU.
- the term *service* is used in the meaning defined by the **service-oriented architecture** (SOA) [3]. This meaning has the strongest relation to the usage of the term *service* on the *AUTOSAR adaptive platform*.

1.3 Abbreviations

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

Abbreviation	*Meaning*
AES	Advanced Encryption Standard
API	Application Programming Interface
ATP	AUTOSAR Template Profile
ARXML	AUTOSAR XML
CAN	Controller Area Network
CRC	Cyclic Redundancy Check
CTM	Counter Mode
DDS	Data Distribution Service
DES	Data Encryption Standard
DHCP	Dynamic Host Control Protocol
DoIP	Diagnostics over IP
DM	Diagnostic Manager
DTC	Diagnostic Trouble Code
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECU	Electrical Control Unit
ECIES	Elliptic Curve Integrated Encryption Scheme
EDDSA	Edwards-Curve Digital Signature Algorithm
FQDN	Fully-Qualified Domain Name
GCM	Galios/Counter Mode
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transport Protocol
ID	Identifier
IO	Input/Output
IP	Internet Protocol
ISO	International Standardization Organization
JSON	JavScript Object Notation
LAN	Local Area Network
MAC	Media Access Control
MAC	Message Authentication Code
MD	Message Digest
MTU	Maximum Transmission Unit
NM	Network Management
NV	Non-Volatile
OEM	Original Equipment Manufacturer

▽

△

Abbreviation	*Meaning*
OS	Operating System
PDU	Protocol Data Unit
PHM	Platform Health Management
PKCS	Public Key Cryptography Standards
POSIX	Portable Operating System Interface
PSK	Pre-Shared Key
RAM	Random Access Memory
REST	Representational State Transfer
ROM	Read-Only Memory
RSA	Cryptographic approach according to Rivest, Shamir, and Adleman
SD	Service Discovery
SDG	Special Data Group
SHA	Secure Hash Algorithm
SOME/IP	Scalable service-Oriented MiddlewarE over IP
SWC	Software Component
TCP	Transport Control Protocol
TLS	Transport Layer Security
TLV	Tag Length Value
TTL	Time to Live
UDS	Unified Diagnostic Services
UDP	User datagram Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VFB	Virtual Functional Bus
VLAN	Virtual Local Area Network
VSA	Variable Size Array
XML	Extensible Markup Language
XSD	XML Schema Definition

Table 1.1: Abbreviations used in the scope of this Document

1.4 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:

Class	**AUTOSAR**			
Package	M2::AUTOSARTemplates::AutosarTopLevelStructure			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. **Tags:** xml.globalElement=true			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Tags:** xml.sequenceOffset=10
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
fileInfo Comment	FileInfoComment	0..1	aggr	This represents a possibility to provide a structured comment in an AUTOSAR file. **Stereotypes:** atpStructuredComment **Tags:** xml.roleElement=true xml.sequenceOffset=-10 xml.typeElement=false
introduction	DocumentationBlock	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to rpresent disclaimers and legal notes. **Tags:** xml.sequenceOffset=20

Table 1.2: AUTOSAR

The first rows in the table have the following meaning:

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

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

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

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

The headers in the table have the following meaning:

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

Type: The type of an attribute of the class.

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

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

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

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

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([4]).

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([4]).

1.5 Requirements Tracing

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

The following table 1.3 references the requirements specified in the corresponding requirements document and provides information about individual specification items that fulfill a given requirement.

Requirement	Description	Satisfied by

[RS_MANI_00002]	Declaration of provided and required services in an application	[TPS_MANI_01039] [TPS_MANI_01040] [TPS_MANI_01052] [TPS_MANI_01053] [TPS_MANI_01057] [TPS_MANI_03210] [TPS_MANI_03211] [TPS_MANI_03212]
[RS_MANI_00003]	Specification of service interfaces	[TPS_MANI_01001] [TPS_MANI_01004] [TPS_MANI_01005] [TPS_MANI_01006] [TPS_MANI_01007] [TPS_MANI_01033] [TPS_MANI_01034] [TPS_MANI_01035] [TPS_MANI_01064] [TPS_MANI_03118] [TPS_MANI_03119] [TPS_MANI_03223]
[RS_MANI_00004]	Support of application design	[TPS_MANI_01010]
[RS_MANI_00005]	Configuration of diagnostic capabilities of an application	[TPS_MANI_01037] [TPS_MANI_01048] [TPS_MANI_01049] [TPS_MANI_01050] [TPS_MANI_01051] [TPS_MANI_01060] [TPS_MANI_01259] [TPS_MANI_01260] [TPS_MANI_01261] [TPS_MANI_01262] [TPS_MANI_01263] [TPS_MANI_01264] [TPS_MANI_01266]
[RS_MANI_00006]	Support of application deployment	[TPS_MANI_01011]
[RS_MANI_00007]	Configuration of application startup behavior	[TPS_MANI_01012] [TPS_MANI_01013] [TPS_MANI_01014] [TPS_MANI_01015] [TPS_MANI_01017] [TPS_MANI_01041] [TPS_MANI_01046] [TPS_MANI_01059] [TPS_MANI_01061] [TPS_MANI_01188] [TPS_MANI_01209]
[RS_MANI_00008]	Service interface deployment to a transport layer mechanism	[TPS_MANI_01136] [TPS_MANI_01137] [TPS_MANI_01210] [TPS_MANI_03036] [TPS_MANI_03037] [TPS_MANI_03038] [TPS_MANI_03039] [TPS_MANI_03070] [TPS_MANI_03071] [TPS_MANI_03072] [TPS_MANI_03073] [TPS_MANI_03074] [TPS_MANI_03075] [TPS_MANI_03101] [TPS_MANI_03103] [TPS_MANI_03104] [TPS_MANI_03105] [TPS_MANI_03106] [TPS_MANI_03107] [TPS_MANI_03108] [TPS_MANI_03116] [TPS_MANI_03117] [TPS_MANI_03217]
[RS_MANI_00009]	Service instance configuration on the network-level	[TPS_MANI_03001] [TPS_MANI_03002] [TPS_MANI_03003] [TPS_MANI_03004] [TPS_MANI_03007] [TPS_MANI_03008] [TPS_MANI_03009] [TPS_MANI_03010] [TPS_MANI_03022] [TPS_MANI_03023] [TPS_MANI_03024] [TPS_MANI_03049] [TPS_MANI_03061] [TPS_MANI_03554] [TPS_MANI_03555]
[RS_MANI_00011]	Instantiation of provided and required services in an application	[TPS_MANI_03000]
[RS_MANI_00014]	User defined transport layer mechanisms	[TPS_MANI_01165] [TPS_MANI_03032] [TPS_MANI_03045] [TPS_MANI_03046] [TPS_MANI_03047] [TPS_MANI_03048] [TPS_MANI_03102]

[RS_MANI_00015]	Definition of the nature of a manifest	[TPS_MANI_01000] [TPS_MANI_01019] [TPS_MANI_01020] [TPS_MANI_01021]
[RS_MANI_00016]	Usage of data types specifically on the AUTOSAR adaptive platform	[TPS_MANI_01016] [TPS_MANI_01027] [TPS_MANI_01047] [TPS_MANI_01100]
[RS_MANI_00017]	Specification of the mapping of Service Interfaces	[TPS_MANI_01002] [TPS_MANI_01003] [TPS_MANI_01022] [TPS_MANI_01024] [TPS_MANI_01025] [TPS_MANI_01026] [TPS_MANI_01032]
[RS_MANI_00018]	Network connections of the machine	[TPS_MANI_03052] [TPS_MANI_03053]
[RS_MANI_00019]	Service discovery message exchange configuration	[TPS_MANI_03064]
[RS_MANI_00020]	Hardware resources of the machine	[TPS_MANI_03035] [TPS_MANI_03065]
[RS_MANI_00021]	Description of machine states	[TPS_MANI_03035]
[RS_MANI_00022]	Adaptive Platform configuration	[TPS_MANI_01208] [TPS_MANI_03035]
[RS_MANI_00023]	Adaptive Module configuration	[TPS_MANI_01208] [TPS_MANI_01226] [TPS_MANI_01227] [TPS_MANI_03035] [TPS_MANI_03056] [TPS_MANI_03096] [TPS_MANI_03098] [TPS_MANI_03162] [TPS_MANI_03163] [TPS_MANI_03164] [TPS_MANI_03165] [TPS_MANI_03166] [TPS_MANI_03167] [TPS_MANI_03218] [TPS_MANI_03219] [TPS_MANI_03220] [TPS_MANI_03221] [TPS_MANI_03222] [TPS_MANI_03226] [TPS_MANI_03502] [TPS_MANI_03503] [TPS_MANI_03505] [TPS_MANI_03506] [TPS_MANI_03508] [TPS_MANI_03509] [TPS_MANI_03510] [TPS_MANI_03511] [TPS_MANI_03512] [TPS_MANI_03513] [TPS_MANI_03514] [TPS_MANI_03515] [TPS_MANI_03516] [TPS_MANI_03517] [TPS_MANI_03518] [TPS_MANI_03519] [TPS_MANI_03520] [TPS_MANI_03521] [TPS_MANI_03522] [TPS_MANI_03523] [TPS_MANI_03524] [TPS_MANI_03544] [TPS_MANI_03545] [TPS_MANI_03546] [TPS_MANI_03552] [TPS_MANI_03553]
[RS_MANI_00024]	SOME/IP transport layer mechanisms	[TPS_MANI_01136] [TPS_MANI_01137] [TPS_MANI_03002] [TPS_MANI_03003] [TPS_MANI_03004] [TPS_MANI_03007] [TPS_MANI_03008] [TPS_MANI_03009] [TPS_MANI_03010] [TPS_MANI_03011] [TPS_MANI_03012] [TPS_MANI_03013] [TPS_MANI_03014] [TPS_MANI_03015] [TPS_MANI_03016] [TPS_MANI_03017] [TPS_MANI_03018] [TPS_MANI_03020] [TPS_MANI_03021] [TPS_MANI_03022] [TPS_MANI_03023] [TPS_MANI_03024] [TPS_MANI_03025] [TPS_MANI_03026]

		[TPS_MANI_03027] [TPS_MANI_03028] [TPS_MANI_03029] [TPS_MANI_03030] [TPS_MANI_03031] [TPS_MANI_03040] [TPS_MANI_03041] [TPS_MANI_03042] [TPS_MANI_03043] [TPS_MANI_03044] [TPS_MANI_03049] [TPS_MANI_03050] [TPS_MANI_03051] [TPS_MANI_03057] [TPS_MANI_03059] [TPS_MANI_03061] [TPS_MANI_03067] [TPS_MANI_03068] [TPS_MANI_03069] [TPS_MANI_03070] [TPS_MANI_03071] [TPS_MANI_03072] [TPS_MANI_03073] [TPS_MANI_03074] [TPS_MANI_03075] [TPS_MANI_03116] [TPS_MANI_03154] [TPS_MANI_03155] [TPS_MANI_03156] [TPS_MANI_03157] [TPS_MANI_03158] [TPS_MANI_03159] [TPS_MANI_03168] [TPS_MANI_03217] [TPS_MANI_03554] [TPS_MANI_03555]
[RS_MANI_00025]	Definition and configuration of serialization	[TPS_MANI_01210] [TPS_MANI_03101] [TPS_MANI_03102] [TPS_MANI_03103] [TPS_MANI_03104] [TPS_MANI_03105] [TPS_MANI_03106] [TPS_MANI_03107] [TPS_MANI_03108] [TPS_MANI_03117]
[RS_MANI_00026]	Software Component System Design	[TPS_MANI_01054] [TPS_MANI_01191] [TPS_MANI_01192] [TPS_MANI_01198] [TPS_MANI_03110] [TPS_MANI_03111] [TPS_MANI_03112] [TPS_MANI_03113] [TPS_MANI_03114] [TPS_MANI_03115]
[RS_MANI_00027]	Support for access to persistent data	[TPS_MANI_01065] [TPS_MANI_01067] [TPS_MANI_01068] [TPS_MANI_01069] [TPS_MANI_01073] [TPS_MANI_01078] [TPS_MANI_01079] [TPS_MANI_01080] [TPS_MANI_01081] [TPS_MANI_01135] [TPS_MANI_01138] [TPS_MANI_01139] [TPS_MANI_01140] [TPS_MANI_01141] [TPS_MANI_01142] [TPS_MANI_01143] [TPS_MANI_01144] [TPS_MANI_01146] [TPS_MANI_01147] [TPS_MANI_01148] [TPS_MANI_01149] [TPS_MANI_01150] [TPS_MANI_01151] [TPS_MANI_01152] [TPS_MANI_01154] [TPS_MANI_01155] [TPS_MANI_01156] [TPS_MANI_01157] [TPS_MANI_01158] [TPS_MANI_01159] [TPS_MANI_01160] [TPS_MANI_01179] [TPS_MANI_01180] [TPS_MANI_01182] [TPS_MANI_01183] [TPS_MANI_01187] [TPS_MANI_01194] [TPS_MANI_01195] [TPS_MANI_01196] [TPS_MANI_01197] [TPS_MANI_01204] [TPS_MANI_01205] [TPS_MANI_01206] [TPS_MANI_01207]
[RS_MANI_00028]	Configuration of Safety protection	[TPS_MANI_03127] [TPS_MANI_03128] [TPS_MANI_03129] [TPS_MANI_03130] [TPS_MANI_03131] [TPS_MANI_03132]

[RS_MANI_00029]	Mapping description between Signal-based communication and Service-Oriented communication	[TPS_MANI_03120] [TPS_MANI_03124] [TPS_MANI_03125] [TPS_MANI_03126]
[RS_MANI_00030]	Definition of optional elements in composite data structures	[TPS_MANI_01097] [TPS_MANI_01184] [TPS_MANI_01185] [TPS_MANI_01186]
[RS_MANI_00032]	Support for platform health management	[TPS_MANI_03500] [TPS_MANI_03502] [TPS_MANI_03503] [TPS_MANI_03505] [TPS_MANI_03506] [TPS_MANI_03508] [TPS_MANI_03509] [TPS_MANI_03510] [TPS_MANI_03511] [TPS_MANI_03512] [TPS_MANI_03513] [TPS_MANI_03514] [TPS_MANI_03515] [TPS_MANI_03516] [TPS_MANI_03517] [TPS_MANI_03518] [TPS_MANI_03519] [TPS_MANI_03520] [TPS_MANI_03521] [TPS_MANI_03522] [TPS_MANI_03523] [TPS_MANI_03524] [TPS_MANI_03534] [TPS_MANI_03544] [TPS_MANI_03545] [TPS_MANI_03546] [TPS_MANI_03552] [TPS_MANI_03553]
[RS_MANI_00033]	Interaction with web services based on the REST pattern	[TPS_MANI_01103] [TPS_MANI_01105] [TPS_MANI_01120] [TPS_MANI_01121] [TPS_MANI_01122] [TPS_MANI_01123] [TPS_MANI_01124] [TPS_MANI_01125] [TPS_MANI_01126] [TPS_MANI_01127] [TPS_MANI_01128] [TPS_MANI_01129] [TPS_MANI_01130] [TPS_MANI_01131] [TPS_MANI_01178]
[RS_MANI_00034]	Specification of capabilities	[TPS_MANI_01106] [TPS_MANI_01107] [TPS_MANI_01108] [TPS_MANI_03209]
[RS_MANI_00035]	Definition of an uploadable software package	[TPS_MANI_01109] [TPS_MANI_01110] [TPS_MANI_01111] [TPS_MANI_01112] [TPS_MANI_01113] [TPS_MANI_01114] [TPS_MANI_01115] [TPS_MANI_01116] [TPS_MANI_01117] [TPS_MANI_01118] [TPS_MANI_01119] [TPS_MANI_01161] [TPS_MANI_01162] [TPS_MANI_01163] [TPS_MANI_01164] [TPS_MANI_01189] [TPS_MANI_01202] [TPS_MANI_01211] [TPS_MANI_01213] [TPS_MANI_01214] [TPS_MANI_01215] [TPS_MANI_01216] [TPS_MANI_01217] [TPS_MANI_01218] [TPS_MANI_01219] [TPS_MANI_01220] [TPS_MANI_01221] [TPS_MANI_01222] [TPS_MANI_01223] [TPS_MANI_01224] [TPS_MANI_01225]
[RS_MANI_00036]	Configuration of security protection	[TPS_MANI_03133] [TPS_MANI_03134] [TPS_MANI_03137] [TPS_MANI_03138] [TPS_MANI_03139] [TPS_MANI_03140] [TPS_MANI_03199] [TPS_MANI_03200] [TPS_MANI_03203] [TPS_MANI_03204] [TPS_MANI_03205] [TPS_MANI_03206] [TPS_MANI_03207] [TPS_MANI_03208] [TPS_MANI_03213] [TPS_MANI_03214] [TPS_MANI_03216]

[RS_MANI_00037]	Configuration of logging and tracing	[TPS_MANI_03160] [TPS_MANI_03161]
[RS_MANI_00038]	DDS transport layer mechanisms	[TPS_MANI_03525] [TPS_MANI_03526] [TPS_MANI_03527] [TPS_MANI_03528] [TPS_MANI_03529] [TPS_MANI_03530] [TPS_MANI_03531] [TPS_MANI_03532] [TPS_MANI_03533] [TPS_MANI_03556] [TPS_MANI_03557] [TPS_MANI_03558] [TPS_MANI_03559] [TPS_MANI_03560] [TPS_MANI_03561] [TPS_MANI_03562] [TPS_MANI_03563] [TPS_MANI_03564] [TPS_MANI_03565] [TPS_MANI_03566] [TPS_MANI_03567] [TPS_MANI_03568] [TPS_MANI_03569] [TPS_MANI_03570] [TPS_MANI_03571] [TPS_MANI_03572]
[RS_MANI_00039]	Usage of implementation specific data types	[TPS_MANI_01166] [TPS_MANI_01167] [TPS_MANI_01168] [TPS_MANI_01169] [TPS_MANI_01171] [TPS_MANI_01172] [TPS_MANI_01173] [TPS_MANI_01174] [TPS_MANI_01175] [TPS_MANI_01176] [TPS_MANI_01177] [TPS_MANI_01201] [TPS_MANI_01212] [TPS_MANI_03169] [TPS_MANI_03170] [TPS_MANI_03171] [TPS_MANI_03172] [TPS_MANI_03173] [TPS_MANI_03174] [TPS_MANI_03175] [TPS_MANI_03176] [TPS_MANI_03177] [TPS_MANI_03178] [TPS_MANI_03179] [TPS_MANI_03180] [TPS_MANI_03181] [TPS_MANI_03183] [TPS_MANI_03184] [TPS_MANI_03185] [TPS_MANI_03186] [TPS_MANI_03187] [TPS_MANI_03188] [TPS_MANI_03189] [TPS_MANI_03190] [TPS_MANI_03191] [TPS_MANI_03192] [TPS_MANI_03193] [TPS_MANI_03196] [TPS_MANI_03197] [TPS_MANI_03198] [TPS_MANI_03201] [TPS_MANI_03202]
[RS_MANI_00040]	Support for access to synchronized time	[TPS_MANI_03535] [TPS_MANI_03536] [TPS_MANI_03537] [TPS_MANI_03538] [TPS_MANI_03539] [TPS_MANI_03540] [TPS_MANI_03541] [TPS_MANI_03542] [TPS_MANI_03543] [TPS_MANI_03547] [TPS_MANI_03548] [TPS_MANI_03549] [TPS_MANI_03550] [TPS_MANI_03551]
[RS_MANI_00041]	Configuration of function groups	[TPS_MANI_03145] [TPS_MANI_03152] [TPS_MANI_03194] [TPS_MANI_03195]
[RS_MANI_00050]	Support of Deterministic Client	[TPS_MANI_01199] [TPS_MANI_01200] [TPS_MANI_01203]
[RS_MANI_00060]	Support of Identity and Access Management	[TPS_MANI_01231] [TPS_MANI_01232] [TPS_MANI_01233] [TPS_MANI_01234] [TPS_MANI_01235] [TPS_MANI_01237] [TPS_MANI_01238] [TPS_MANI_01239] [TPS_MANI_01240] [TPS_MANI_01241]

[RS_MANI_00061]	Support of Diagnostic Interfaces	[TPS_MANI_01048] [TPS_MANI_01049] [TPS_MANI_01050] [TPS_MANI_01051] [TPS_MANI_01242] [TPS_MANI_01243] [TPS_MANI_01245] [TPS_MANI_01246] [TPS_MANI_01247] [TPS_MANI_01248] [TPS_MANI_01249] [TPS_MANI_01250] [TPS_MANI_01251] [TPS_MANI_01252] [TPS_MANI_01253] [TPS_MANI_01254] [TPS_MANI_01255] [TPS_MANI_01256] [TPS_MANI_01257] [TPS_MANI_01258] [TPS_MANI_01259] [TPS_MANI_01260] [TPS_MANI_01261] [TPS_MANI_01262] [TPS_MANI_01263] [TPS_MANI_01264] [TPS_MANI_01265] [TPS_MANI_01266]
[RS_MANI_00062]	Support for Partial Networking	[TPS_MANI_03224] [TPS_MANI_03225]

Table 1.3: RequirementsTracing

1.6 Known Limitations

The AUTOSAR SWS REST [5] defines a low-level API for REST-based communication. The content of section 11, on the other hand, applies for the configuration of a not-yet standardized API on top of the ara::rest API.

2 Big Picture of Manifest Definition

2.1 Design vs. Deployment

2.1.1 Overview

Despite the name, this document contains the description of model elements that are clearly bound to a *design* workflow **and** model elements that have a strong relation to the *deployment* aspect.

Model elements discussed in this document are either related to *design* or *deployment*, there is no overlap between the two groups.

Model elements that are related to *deployment* will be used in models that are uploaded to a target platform, see [TPS_MANI_01000]. These model elements are mainly described in sections of this document where the term “Manifest” is part of the section title.

In the absence of a more precise definition, model elements related to *design* can be identified by not being related to *deployment*.

2.1.2 Relation between Design and Deployment Models

Please note that in many cases the part of the meta-model related to *deployment* reflects a similar modeling in the *design* domain, e.g. the definition of E2E profile parameters.

There is currently no clearly defined preference about how the relation between *design* and *deployment* may impact a concrete development project. The following scenarios for the example of *E2E properties* might occur:

- An OEM delivers the description of `AdaptivePlatformServiceInstance`s including the definition of *E2E properties*.

  It is safe to assume that subsequent processing of the model shall take the *E2E properties* as granted and develop the software with respect to the given properties.

- Software exists that has defined *E2E properties* by means of `ComSpec`s. For various reasons, it may happen that the software cannot be updated and therefore takes the “lead” in terms of the definition of *E2E properties*.

  The definition of `AdaptivePlatformServiceInstance`s may then have to respect the existing modeling on the software side.

- It could also happen that existing definitions can be **partly** overwritten by engineers who **really** know what they are doing.

2.1.3 Structure of the document

The structure of the document maps to the division between *design* and *deployment* such that the *design* aspect is mostly described in sections 3, 4.1 and 11.1.

Chapters 7, 9, 6, 8.2, and 8.3 focus on *deployment*-related content.

2.2 About Manifest

This chapter shall clarify the definition of the term `Manifest` in the context of the *AUTOSAR adaptive platform*.

[TPS_MANI_01000]{DRAFT} **Definition of the term `Manifest`** ⌈ A `Manifest` represents a piece of AUTOSAR model description that is created to support the configuration of an *AUTOSAR adaptive platform* product and which is uploaded to the *AUTOSAR adaptive platform* product, potentially in combination with other artifacts (like binary files) that contain executable code to which the `Manifest` applies. ⌋ *(RS_MANI_00015)*

It is important to stress the fact that the usage of a `Manifest` is indeed strictly limited to the *AUTOSAR adaptive platform* and that there is no use case to port the concept to the *AUTOSAR classic platform*.

2.3 Serialization Format

One aspect that the definition of a `Manifest` has in common with other AUTOSAR model content is the standardized serialization format.

[TPS_MANI_01020]{DRAFT} **Serialization format of the `Manifest` in AUTOSAR** ⌈ The standardized serialization format of `Manifest` content in AUTOSAR is ARXML.

Consequently, `Manifest` model content can be validated against the AUTOSAR XML Schema. ⌋*(RS_MANI_00015)*

An important consequence of [TPS_MANI_01020] is that there is no limitation to just one "manifest file" a.k.a. "the manifest".

Content may be distributed among several physical files according to the rules given in the specification of the AUTOSAR Generic Structure Template [6].

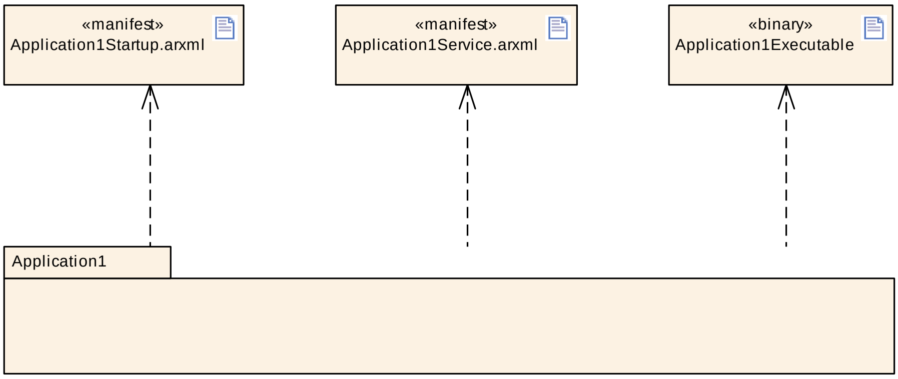


Figure 2.1: Example usage of several manifest files within one software delivery

[TPS_MANI_01021]{DRAFT} **Serialization format of Manifest content on a machine** ⌈ The serialization format used to actually upload a manifest on a machine may be freely chosen by a platform supplier.

However, the content and semantics of the original ARXML Manifest needs to be **fully preserved**. ⌋*(RS_MANI_00015)*

It can be expected that in many cases the best option for the upload of the Manifest will still be ARXML because a custom format obviously has to support the full complexity of the Manifest meta-model.

Please note that the meta-model foresees the existence of references from manifest-related meta-classes to design-related meta-classes.

These references are created for the sake of clarity but it is not mandatory that the content of the reference actually needs to be resolvable.

In terms of the AUTOSAR modeling approach, this translates to a decoration of these references with the stereotype ≪atpUriDef≫. More information can be found in [6].

If the referenced meta-classes contain information that is relevant for the manifest level then this information is replicated on the manifest level (such that the manifest-level model does not have to rely on the availability of design-level information).

2.4 Scope

As mentioned before, the usage of a Manifest is limited to the *AUTOSAR adaptive platform*. This does not mean, however, that all ARXML produced in a development project that targets the *AUTOSAR adaptive platform* is automatically considered a Manifest.

In fact, the *AUTOSAR adaptive platform* is usually not exclusively used in a vehicle project.

A typical vehicle will most likely be also equipped with a number of ECUs developed on the *AUTOSAR classic platform* and the system design for the entire vehicle will therefore have to cover both ECUs built on top of the *AUTOSAR classic platform* and those created on top of the *AUTOSAR adaptive platform*.

[TPS_MANI_01019]{DRAFT} **`Manifest` content may apply to different aspects of the *AUTOSAR adaptive platform*** ⌈ `Manifest` content can apply to different aspects of the model. At the moment, `Manifest` content can roughly be divided into three focus areas:

- Application-related `Manifest` content describes all aspects of the deployment of an application, including - but not limited to - the startup configuration and the configuration of service-oriented communication endpoints on application level.
- Machine-related `Manifest` content describes the deployment of just a machine, i.e. without any application (including platform modules) running on the machine.
- Service instance-related `Manifest` describes how service-oriented communication on transport layer level is bound to endpoints in the application and (in some cases) platform software.

⌋*(RS_MANI_00015)*

2.5 Manifests described in this Document

In principle, the term `Manifest` could be defined such that there is conceptually just one "manifest" and every deployment aspect would be handled in this context.

This does not seem appropriate because it became apparent that manifest-related model-elements exist that are relevant in entirely different phases of a typical development project.

This aspect is taken as the main motivation to subdivide the definition of the term `Manifest` in three different partitions:

Execution Manifest This kind of `Manifest` is used to specify the deployment-related information of applications running on the *AUTOSAR adaptive platform*.

An `Execution Manifest` is bundled with the actual executable code in order to support the integration of the executable code onto the machine.

Please find more information regarding this topic in section 7.

Service Instance Manifest This kind of `Manifest` is used to specify how service-oriented communication is configured in terms of the requirements of the underlying transport protocols.

A `Service Instance Manifest` is bundled with the actual executable code that implements the respective usage of service-oriented communication.

Please find more information regarding this topic in section 9.

Machine Manifest This kind of Manifest is supposed to describe deployment-related content that applies to the configuration of just the underlying machine (i.e. without any applications running on the machine) that runs an *AUTOSAR adaptive platform*.

A Machine Manifest is bundled with the software taken to establish an instance of the *AUTOSAR adaptive platform*.

Please find more information regarding this topic in section 6.

The temporal division between the definition (and usage) of different kinds of Manifest leads to the conclusion that in most cases different physical files will be used to store the content of the three kinds of Manifest.

However, as with all kinds of ARXML content, this is not a binding rule.

3 Application Design

3.1 Overview

This chapter describes all design-related modeling that applies to the creation of application software on the *AUTOSAR adaptive platform*.

This also extends to extensions of existing modeling used on the *AUTOSAR classic platform*, e.g. the introduction of new values of the attribute `category`.

In particular, this section of the document focuses on the following aspects:

- Definition of a dedicated subclass of `SwComponentType` for the *AUTOSAR adaptive platform* (section 3.2)
- Definition of data types specifically for the *AUTOSAR adaptive platform* (section 3.3)
- Service interface as the pivotal element for service-oriented communication (section 3.4)
- Service interface mapping as a mediator between internal and external communication (section 3.5)
- Service interface **element** mapping as a mediator between internal and external communication (section 3.6)
- Persistency interface as the basis for interacting with persistent data storage (section 3.8)
- Aspects of the fine-grained configuration of interaction with the “outside world” from the perspective of the inside of a software-component (section 3.12)
- `Executable` as the smallest executable unit (section 3.13)
- Configuration of transformation properties (section 3.15)

3.2 Software Component

In principle, it would be possible to directly take over the definition of e.g. `ApplicationSwComponentType` for the usage on the *AUTOSAR adaptive platform*.

However, this would complicate the formulation of constraints regarding the existence of model elements (for example: data types, as explained in section 3.3) that are exclusive to the *AUTOSAR adaptive platform*.

Therefore, the `AdaptiveApplicationSwComponentType` is defined as a representation of software-components on the *AUTOSAR adaptive platform*.

The Existence of the AdaptiveApplicationSwComponentType allows for a convenient way (see [constr_1492]) to lock out most kinds of software-component defined for the *AUTOSAR classic platform* from the usage on the *AUTOSAR adaptive platform*.

The clarification of the opposite direction (i.e. an erroneous use of an AdaptiveApplicationSwComponentType) is less obvious.

In other words, it may be possible to use an AdaptiveApplicationSwComponentType within a System as some sort of overall design model for software on both the *AUTOSAR classic platform* **and** the *AUTOSAR adaptive platform*.

This aspect, however, is not clarified so far nor is a restriction in place that prohibits AdaptiveApplicationSwComponentType to appear in the context of a System.

Later versions of this specification may fix the missing regulation.

Class	**AdaptiveApplicationSwComponentType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to support the formal modeling of application software on the AUTOSAR adaptive platform. Consequently, it shall only be used on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=AdaptiveApplicationSwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
internalBehavior	AdaptiveSwcInternalBehavior	0..1	aggr	This aggregation represents the internal behavior of the AdaptiveApplicationSwComponentType for the AUTOSAR adaptive platform. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior, variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=preCompileTime

Table 3.1: AdaptiveApplicationSwComponentType

Class	**AdaptiveSwcInternalBehavior**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::AdaptiveInternalBehavior			
Note	This meta-class represents the ability to define an internal behavior of an AtomicSwComponentType used on the AUTOSAR adaptive platform. Please note that the model of internal behavior in this case, in stark contrast to the situation of the AUTOSAR classic platform, is very minimal. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
serviceDependency	SwcServiceDependency	*	aggr	This represents the collection of SwcServiceDependencys owned by AdaptiveInternalBehavior. **Tags:** atp.Status=draft

Table 3.2: AdaptiveSwcInternalBehavior

3.3 Data Type

3.3.1 Overview

The specification of data types on the *AUTOSAR adaptive platform* follows the same pattern as the counterpart on the *AUTOSAR classic platform*: data types are defined on different levels of abstraction that complement each other.

In the context of this document, the focus is on the discussion of `ApplicationDataType`s and `CppImplementationDataType`s.

In general, most of the concepts regarding the definition of data types can be taken over from the existing specifications on the *AUTOSAR classic platform*.

However, some aspects are specific to the *AUTOSAR adaptive platform* and are consequently discussed in the scope of this document rather than the specification of the AUTOSAR Software Component Template [1].

One of the aspects that could be taken over from the *AUTOSAR classic platform* is the definition of initial values.

Although the utility of initial values is certainly limited on the *AUTOSAR adaptive platform*, there is an opportunity to utilize the definition of initial values in the context of the so-called `Field`s (see [TPS_MANI_01034]).

3.3.2 ApplicationDataType

The full range of the modeling of `ApplicationDataType`s that is supported on the *AUTOSAR classic platform* can directly be used on the *AUTOSAR adaptive platform* as well.

In addition to the `ApplicationDataType`s supported on the *AUTOSAR classic platform*, there are further `ApplicationDataType`s that - while in principle also available on the *AUTOSAR classic platform* - are primarily used on and designed for the *AUTOSAR adaptive platform*.

Class	***ApplicationDataType*** (abstract)
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes
Note	ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc. It should be possible to model the application level aspects of a VFB system by using ApplicationData Types only.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	ApplicationDataType (abstract)			
Subclasses	ApplicationCompositeDataType, ApplicationPrimitiveDataType			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 3.3: ApplicationDataType

3.3.2.1 String Data Type

While the handling of data types that represent textual strings is very similar with respect to the definition of ApplicationDataTypes on the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*, special regulations apply on the level of CppImplementationDataTypes on the *AUTOSAR adaptive platform*.

For more information about the modeling of string data types on the level of CppImplementationDataType please refer to section 3.3.3.4.

For the sake of consistency, this chapter summarizes the modeling of ApplicationDataTypes for the modeling of data types that represent textual strings as far as the *AUTOSAR adaptive platform* is concerned.

The meta-classes used to define an ApplicationPrimitiveDataType of category STRING are summarized in Figure 3.1.

Please note that thanks to the usage of programming languages with richer data types than plain C, the implementation of an ApplicationPrimitiveDataType of category STRING on the *AUTOSAR adaptive platform* is predefined for a given *language binding*.

[TPS_MANI_01047]{DRAFT} **Existence of SwRecordLayout for an ApplicationPrimitiveDataType of category STRING** ⌈ For the usage of an ApplicationPrimitiveDataType of category STRING on the *AUTOSAR adaptive platform*, the existence of ApplicationPrimitiveDataType.swDataDefProps.swRecordLayout shall be ignored. ⌋*(RS_MANI_00016)*

Please note that [TPS_MANI_01047] intentionally does not forbid the existence of SwRecordLayout because the same ApplicationPrimitiveDataType of category STRING could rightfully be used **on both** the *AUTOSAR adaptive platform* and the *AUTOSAR classic platform*.

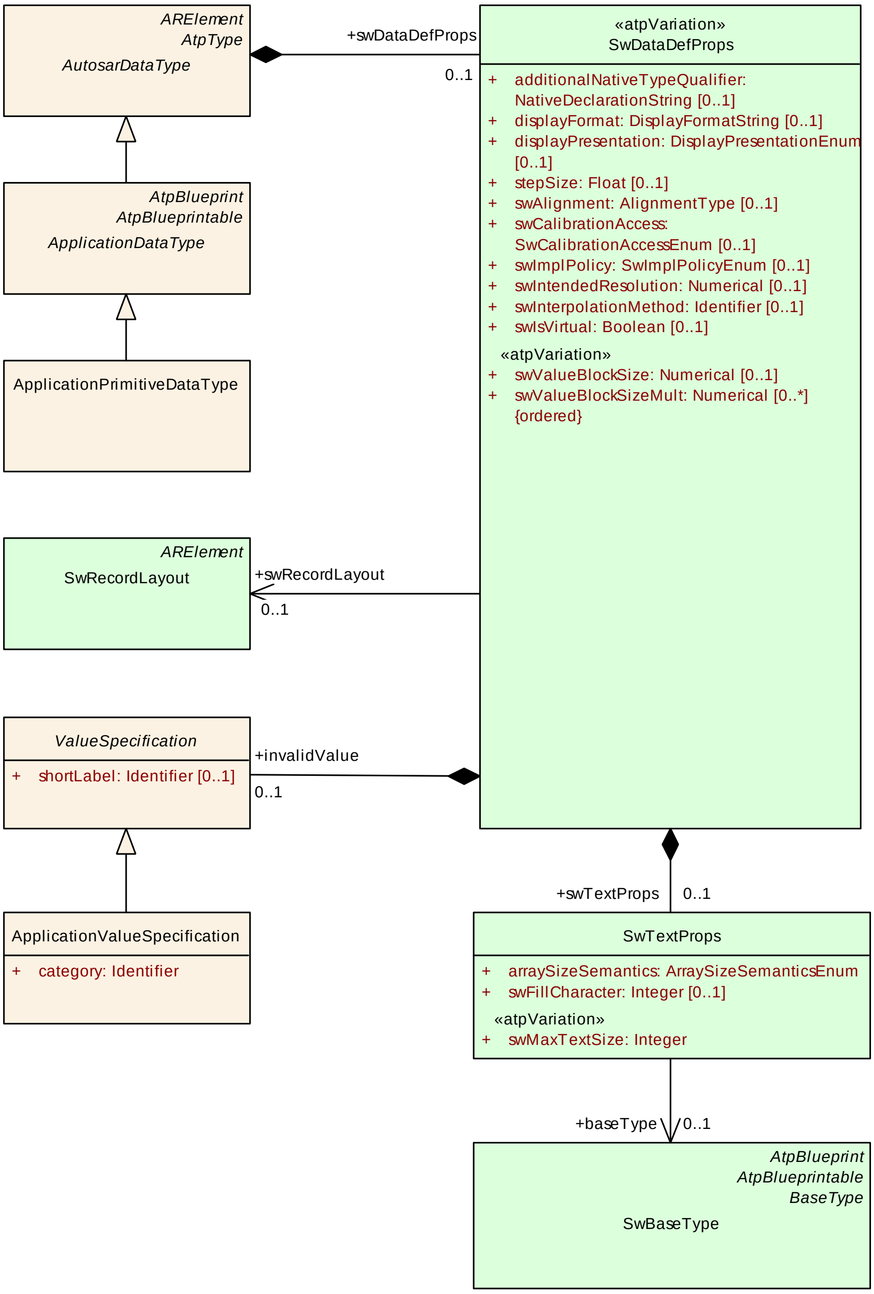


Figure 3.1: Specification of textual strings

Class	**ApplicationPrimitiveDataType**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes
Note	A primitive data type defines a set of allowed values. **Tags:** atp.recommendedPackage=ApplicationDataTypes
Base	*ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	ApplicationPrimitiveDataType			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.4: ApplicationPrimitiveDataType

Class	SwTextProps			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This meta-class expresses particular properties applicable to strings in variables or calibration parameters.			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
arraySize Semantics	ArraySizeSemantics Enum	1	attr	This attribute controls the semantics of the arraysize for the array representing the string in an Implementation DataType. It is there to support a safe conversion between ApplicationDatatype and ImplementationDatatype, even for variable length strings as required e.g. for Support of SAE J1939.
baseType	SwBaseType	0..1	ref	This is the base type of one character in the string. In particular this baseType denotes the intended encoding of the characters in the string on level of ApplicationData Type. **Tags:** xml.sequenceOffset=30
swFillCharacter	Integer	0..1	attr	Filler character for text parameter to pad up to the maximum length swMaxTextSize. The value will be interpreted according to the encoding specified in the associated base type of the data object, e.g. 0x30 (hex) represents the ASCII character zero as filler character and 0 (dec) represents an end of string as filler character. The usage of the fill character depends on the arraySize Semantics. **Tags:** xml.sequenceOffset=40
swMaxTextSize	Integer	1	attr	Specifies the maximum text size in characters. Note the size in bytes depends on the encoding in the corresponding baseType. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

Table 3.5: SwTextProps

3.3.2.2 Associative Map Data Type

[TPS_MANI_01027]{DRAFT} **Semantics of `ApplicationAssocMapDataType`** ⌈
An `ApplicationAssocMapDataType` represents an associative data structure, i.e. a data structure where so-called *keys* (formalized as `ApplicationAssocMapDataType.key` that are in turn typed by an `ApplicationDataType`) are associated with *values* (formalized as `ApplicationAssocMapDataType.value` that are also in turn typed by an `ApplicationDataType`). ⌋*(RS_MANI_00016)*

[constr_3349]{DRAFT} **Usage of `ApplicationAssocMapDataType` is limited** ⌈ The usage of an ApplicationAssocMapDataType is limited to the context of AdaptiveApplicationSwComponentTypes and CompositionSwComponentTypes defined in the context of an Executable, i.e. such a data type shall not be used on the *AUTOSAR classic platform*. ⌋*()*

[constr_3349] is a formal approach to express that an ApplicationAssocMapDataType shall only be used on the *AUTOSAR adaptive platform*.

[TPS_MANI_01016]{DRAFT} **Category of `ApplicationAssocMapDataType`** ⌈ The value ApplicationAssocMapDataType.category shall be set to ASSOCIATIVE_MAP for attribute. ⌋*(RS_MANI_00016)*

Figure 3.2 depicts an example of the structure of an ApplicationAssocMapDataType.

As can be deduced from looking at Figure 3.2, the concept of an ApplicationDataType of category MAP shall not be confused with an ApplicationAssocMapDataType[1].

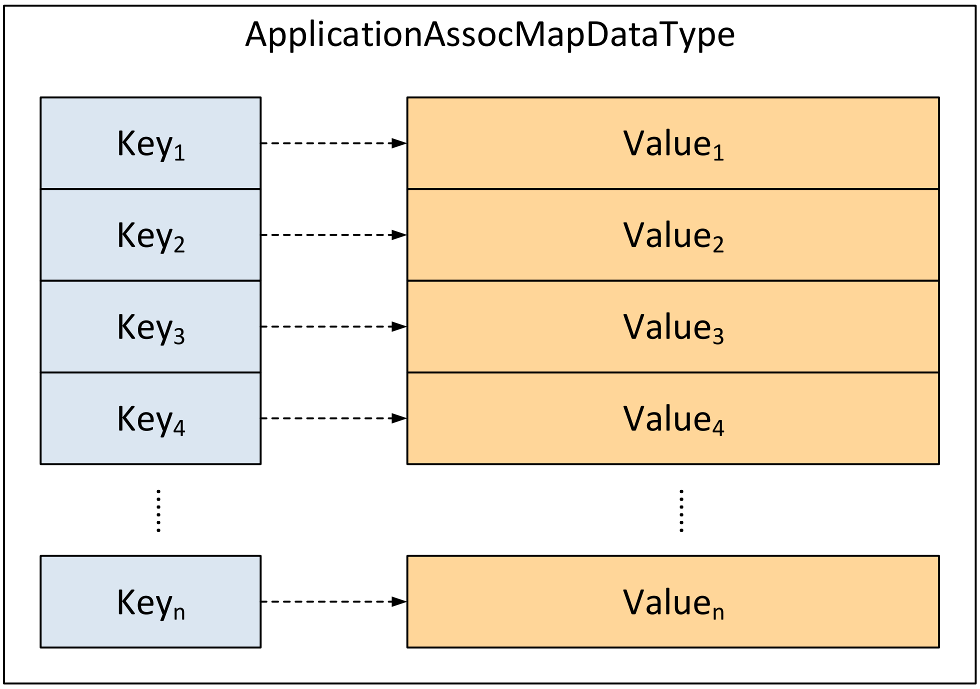


Figure 3.2: Example `ApplicationAssocMapDataType` on the *AUTOSAR adaptive platform*

[1]On the other hand, both concepts of a "map" are justified in their respective "community" and choosing to name one of these very different in order so reduce overall potential confusion would probably not be applicable

There are a number of technical implications on the usage of an associative data structure at run-time, e.g. that the content of each *key* shall be unique within the context of the overall data structure.

On the other hand, it is totally no problem if content on the value-side contain duplicates, e.g. two unique keys are associated with values that have a completely identical content.

However, these aspects have no implication on the formal model of the ApplicationAssocMapDataType and are therefore not considered in this document.

The modeling of the ApplicationAssocMapDataType is somewhat minimalistic and motivated mainly be the fact that data types for both key and value need to be defined.

There is no assumption how the structure of an implementation of an associative map may look like. For example, in C++ (which is currently the only supported language binding on the *AUTOSAR adaptive platform*) the straightforward way to use an associative map is to utilize the container `ara::core::Map` (where the implementation is opaque to the client programmer).

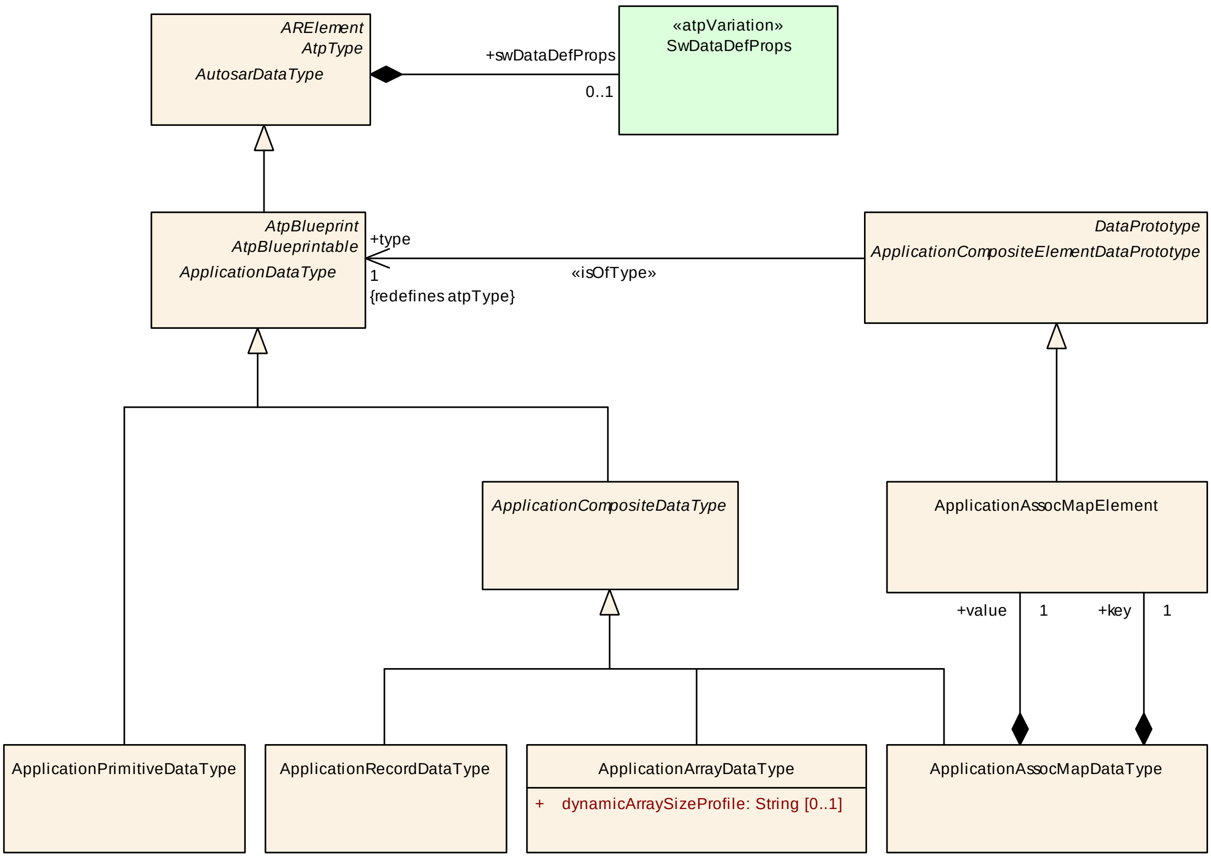


Figure 3.3: Formal model of ApplicationAssocMapDataType

Class	ApplicationAssocMapDataType			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	An application data type which is a map and consists of a key and a value **Tags:** atp.Status=draft atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
key	ApplicationAssocMapElement	1	aggr	Key element of the map that is used to uniquely identify the value of the map. **Tags:** atp.Status=draft
value	ApplicationAssocMapElement	1	aggr	Value element of the map that stores the content associated to a key. **Tags:** atp.Status=draft

Table 3.6: ApplicationAssocMapDataType

Class	ApplicationAssocMapElement			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	Describes the properties of the elements of an application map data type. **Tags:** atp.Status=draft			
Base	*ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.7: ApplicationAssocMapElement

Listing 3.1 provides a sketch of the modeling of an example `ApplicationAssocMapDataType`.

Figure 3.4 contains the corresponding graphical representation of the model.

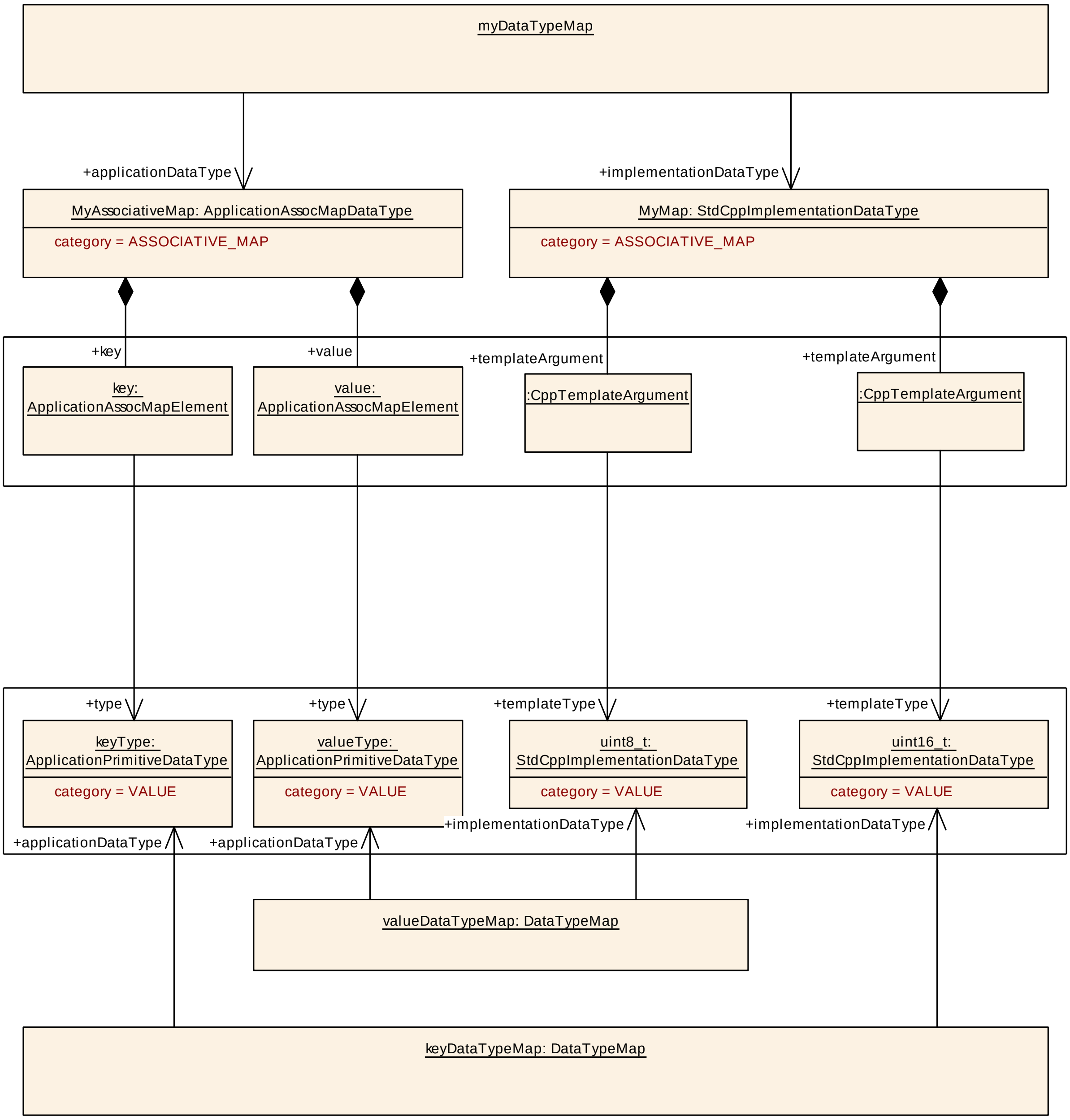


Figure 3.4: Example of the model of an associative map

Listing 3.1: Example for the definition of an ApplicationAssocMapDataType

```
<APPLICATION-ASSOC-MAP-DATA-TYPE>
  <SHORT-NAME>MyAssociativeMap</SHORT-NAME>
  <KEY>
    <SHORT-NAME>MyKey</SHORT-NAME>
    <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">keyType</TYPE-TREF>
  </KEY>
  <VALUE>
    <SHORT-NAME>MyValue</SHORT-NAME>
    <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">valueType</TYPE-TREF>
  </VALUE>
</APPLICATION-ASSOC-MAP-DATA-TYPE>
```

```
<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>keyType</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</APPLICATION-PRIMITIVE-DATA-TYPE>

<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>valueType</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</APPLICATION-PRIMITIVE-DATA-TYPE>
```

The initialization of an ApplicationAssocMapDataType, however, needs to be clarified because it would (using a combination of RecordValueSpecification and ArrayValueSpecification) in general be technically possible to define a number of differently structured ValueSpecifications that are semantically identical.

In order to keep this element of uncertainty out of the AUTOSAR standard, the initialization of a DataPrototype typed by ApplicationAssocMapDataType is clarified by means of [constr_1488].

[constr_1488]{DRAFT} **Initialization of a DataPrototype typed by an ApplicationAssocMapDataType** ⌈ A DataPrototype typed by an ApplicationAssocMapDataType shall only be initialized by an ApplicationAssocMapValueSpecification. ⌋*()*

As already mentioned, there is a semantic requirement that the *key* elements of an *associative map* need to the unique in the context of one *associative map* container.

Obviously, the model has no influence on what happens at run-time. On the other hand, there is an implication onto the initialization of an ApplicationAssocMapDataType, see [constr_1489].

[constr_1489]{DRAFT} **Uniqueness of ApplicationAssocMapValueSpecification.mapElementTuple.key** ⌈ The value of all mapElementTuple.key elements in the context of a given ApplicationAssocMapValueSpecification shall be unique. ⌋*()*

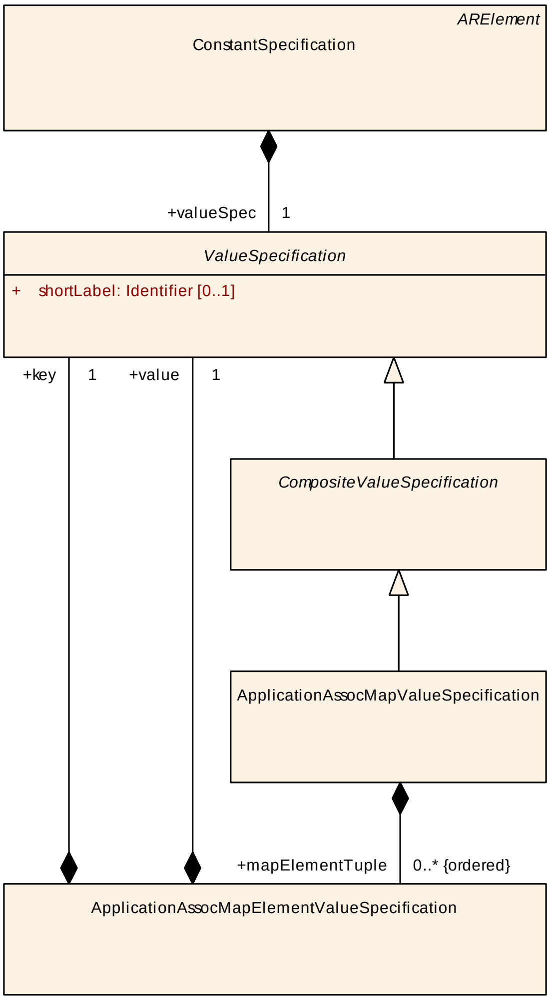


Figure 3.5: Formal model of the initialization of an `ApplicationAssocMapDataType`

Class	**ApplicationAssocMapValueSpecification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	This meta-class represents the ability to define the initialization of an ApplicationAssocMapDataType. **Tags:** atp.Status=draft			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
mapElementTuple (ordered)	ApplicationAssocMapElementValueSpecification	*	aggr	This aggregation represents the initial values for the elements of the ApplicationAssocMapValueSpecification. **Tags:** atp.Status=draft

Table 3.8: ApplicationAssocMapValueSpecification

Class	**ApplicationAssocMapElementValueSpecification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	This meta-class represents the ability to define the initialization of the elements of an ApplicationAssoc MapDataType. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
key	ValueSpecification	1	aggr	This aggregation represents the initialization of the key part of an AssociativeElementValueSpecification. **Tags:** atp.Status=draft
value	ValueSpecification	1	aggr	This aggregation represents the initialization of the value part of an AssociativeElementValueSpecification. **Tags:** atp.Status=draft

Table 3.9: ApplicationAssocMapElementValueSpecification

3.3.2.3 Attributes of SwDataDefProps

[constr_1478]{DRAFT} **`SwDataDefProps` applicable to `ApplicationDataType`s exclusive to the *AUTOSAR adaptive platform*** ⌈ A complete list of the `SwDataDefProps` and other attributes and their multiplicities which are allowed for a given `category` is shown in table 3.10. ⌋*()*

A consequence of [constr_1478] is that the Table 3.10 shows only the values of `category` that are limited to the *AUTOSAR adaptive platform*. For all other values of `category` that are also supported on the *AUTOSAR classic platform* please refer to a similar table contained in the specification of the Software Component Template [1].

Attributes of SwDataDefProps	Root Elem.		Attribute Existence per Category
	ApplicationAssocMapDataType	ApplicationAssocMapElement	ASSOCIATIVE_MAP
additionalNativeTypeQualifier			
annotation	x	x	*
baseType			
compuMethod			
dataConstr			
displayFormat	x	x	0..1
implementationDataType			
invalidValue			
stepSize			
swAddrMethod			
swAlignment			
swBitRepresentation			
swCalibrationAccess			
swCalprmAxisSet			
swComparisonVariable			
swDataDependency			
swHostVariable			
swImplPolicy			
swIntendedResolution			
swInterpolationMethod			
swIsVirtual			
swPointerTargetProps			
swRecordLayout			
swRefreshTiming			
swTextProps			
swValueBlockSize			
unit			
valueAxisDataType			
Other Attributes below the Root Element			
key: ApplicationAssocMapElement	x		1
value: ApplicationAssocMapElement	x		1

Table 3.10: Allowed Attributes vs. category for ApplicationDataTypes

3.3.3 CppImplementationDataType

3.3.3.1 Overview

In the AUTOSAR standard, data types represent assets of paramount prominence for the entire development approach.

Therefore[2], AUTOSAR implements a multi-level approach for the modeling of data types. One of the described levels, the so-called *Implementation Data Level* aims at a modeling on a level that could be described as “language binding” in the parlor of the *AUTOSAR adaptive platform*.

For the *AUTOSAR classic platform*, the *Implementation Data Level* has been addressed by the creation of the `ImplementationDataType` that specifically aims at covering the data type behavior of the C programming language.

In contrast to the *AUTOSAR classic platform*, the *AUTOSAR adaptive platform* currently does not foresee the usage of the C language and instead (at least for the foreseeable future) defines language binding to the C++ language.

It is therefore necessary to provide a modeling approach on the *Implementation Data Level* with a proper support for the capabilities of the C++ language.

While it would technically be feasible to extend the semantics of `ImplementationDataType` for a support of a C++ language binding this would significantly water down the clarity and expressiveness of `ImplementationDataType`[3].

It therefore seems reasonable to add an additional system of meta-classes that specifically supports the usage of data types with an intended binding to the C++ language.

[TPS_MANI_01166]{DRAFT} **Semantics of `CppImplementationDataType`** ⌈ The abstract meta-class `CppImplementationDataType` supports the modeling of data types specifically tailored towards a support for a C++ language binding. ⌋ *(RS_MANI_00039)*

[TPS_MANI_03197]{DRAFT} **Semantics of `StdCppImplementationDataType`** ⌈ Meta-class `StdCppImplementationDataType` supports the modeling of data types that will be mapped to C++ Standard Library features in the C++ language binding. ⌋ *(RS_MANI_00039)*

Please note that Structures (`category` = `STRUCTURE`) and type aliases (`category` = `TYPE_REFERENCE`) are also modeled as `StdCppImplementationDataType`s for simplification reasons.

[TPS_MANI_03198]{DRAFT} **Semantics of `CustomCppImplementationDataType`** ⌈ Meta-class `CustomCppImplementationDataType` supports the

[2] As explained in [1]

[3] And even if it were possible to extend `ImplementationDataType` towards a more or less clean support for C++ it may happen that further language bindings are added to the *AUTOSAR adaptive platform* for which further and further extensions of `ImplementationDataType` would be required.

modeling of data types that will mapped to a custom implementation in the C++ language binding that is declared in the headerFile. ⌋(RS_MANI_00039)

Please note that the category values for a CustomCppImplementationDataType are restricted by [constr_1578].

This means that the modeling of primitive data types and strings is only possible with StdCppImplementationDataTypes. The reason is that the serialization rules that are defined in AUTOSAR for SOME/IP and DDS are based on the defined types of the standard library.

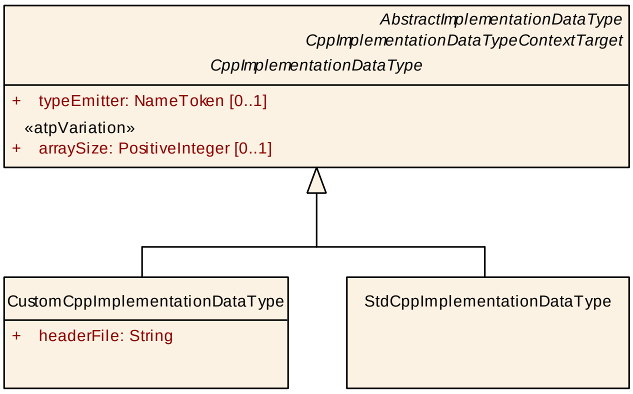


Figure 3.6: Specializations of CppImplementationDataType

Class	***CppImplementationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	CustomCppImplementationDataType, StdCppImplementationDataType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
arraySize	PositiveInteger	0..1	attr	This attribute can be used to specify the array size if the enclosing CppImplementationDataType has array semantics. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CppImplementationData Type. **Tags:** atp.Status=draft
subElement (ordered)	CppImplementation DataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType **Tags:** atp.Status=draft

▽

△

Class	*CppImplementationDataType* (abstract)			
templateArgument (ordered)	CppTemplateArgument	*	aggr	This aggregation allows for the specification of properties of template arguments **Tags:** atp.Status=draft
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.
typeReference	CppImplementation DataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef). **Tags:** atp.Status=draft

Table 3.11: CppImplementationDataType

Class	**StdCppImplementationDataType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a data type definition that is taken as the basis for a C++ language binding to a C++ Standard Library feature. **Tags:** atp.Status=draft atp.recommendedPackage=CppImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataType*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.12: StdCppImplementationDataType

Class	**CustomCppImplementationDataType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a data type definition that is taken as the basis for a C++ language binding to a custom implementation that is declared in the configured header file. The Short Name of this CustomCppImplementationDataType defines the Class-Name of the custom implementation. **Tags:** atp.Status=draft atp.recommendedPackage=CppImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataType*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
headerFile	String	1	attr	Configuration of the Header File with the custom class declaration.

Table 3.13: CustomCppImplementationDataType

[constr_1571]{DRAFT} **`CppImplementationDataType` is limited** ⌈ The usage of a `CppImplementationDataType` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s defined in the context of an `Executable`. ⌋*()*

[TPS_MANI_01167]{DRAFT} **`AbstractImplementationDataType`** ⌈ Meta-class `CppImplementationDataType` inherits from abstract base class `AbstractImplementationDataType` in order to become a valid target for specific references from

other meta-classes that want to refer to "ImplementationDataType in general". ⌋
(*RS_MANI_00039*)

Class	***AbstractImplementationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	This meta-class represents an abstract base class for different flavors of ImplementationDataType.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*CppImplementationDataType*, ImplementationDataType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.14: AbstractImplementationDataType

A prominent example for the idea of referring to "ImplementationDataType in general" can be found in meta-class DataTypeMap. The intention behind the existence of DataTypeMap is to map an ApplicationDataType to either an ImplementationDataType or CppImplementationDataType.

By means of modeling the reference DataTypeMap.implementationDataType as a reference to AbstractImplementationDataType both options are possible in a single role.

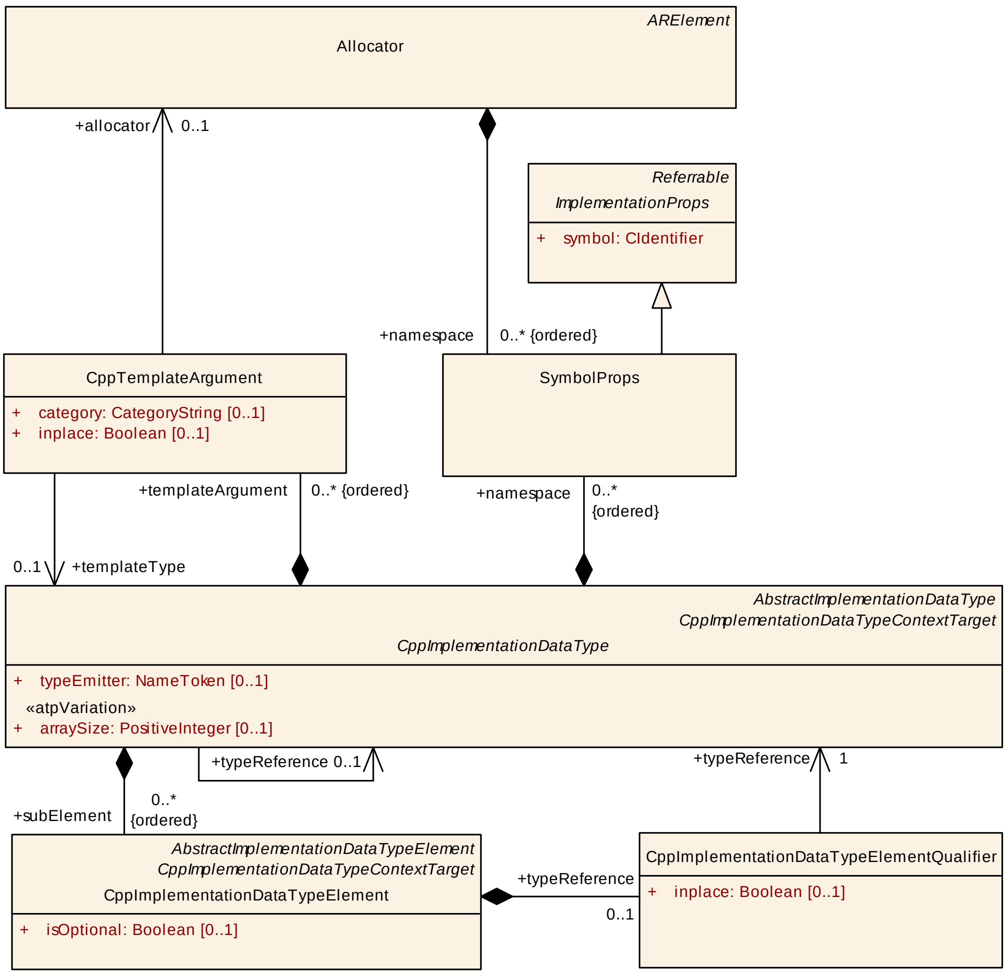


Figure 3.7: CppImplementationDataType overview

In contrast to the C language, C++ supports the definition of namespaces in programs. This feature is also cleared for development on the *AUTOSAR adaptive platform* and therefore needs to be represented in the modeling approach.

[TPS_MANI_01168]{DRAFT} **Specification of a namespace for a CppImplementationDataType** ⌈ The ability to define a namespace for a CppImplementationDataType is expressed by means of the aggregation of SymbolProps at CppImplementationDataType in the role namespace. ⌋*(RS_MANI_00039)*

[constr_3443]{DRAFT} **Specification of a namespace for a StdCppImplementationDataType** ⌈ The definition of a namespace for a StdCppImplementationDataType of category VALUE is not allowed. For this value of category the std namespace is already assumed by the usage of the StdCppImplementationDataType. ⌋*()*

[TPS_MANI_01176]{DRAFT} **Standardized value for attribute CppImplementationDataType.typeEmitter** ⌈ The AUTOSAR Standard reserves the following value for attribute CppImplementationDataType.typeEmitter:

- TYPE_EMITTER_ARA

On top of that, AUTOSAR reserves the standardized names of typical header files for data type definition as further possible values of CppImplementationDataType.typeEmitter (e.g., Platform_Types.h for the CppImplementationDataTypes defined in the Specification of Platform Types for the Adaptive Platform [7]). ⌋*(RS_MANI_00039)*

[TPS_MANI_01177]{DRAFT} **Semantics of CppImplementationDataType.typeEmitter** ⌈ The following set of rules applies for the usage of the attribute CppImplementationDataType.typeEmitter:

- If the attribute typeEmitter is set to the value TYPE_EMITTER_ARA the ARA generator shall generate the corresponding data type definition.
- If the attribute typeEmitter is set to any value other than TYPE_EMITTER_ARA the ARA generator shall silently **not** generate the corresponding data type definition.
- For CppImplementationDataTypes that correspond to fundamental data types float (see [SWS_APT_00043]), double (see [SWS_APT_00046]), and bool (see [SWS_APT_00049]) the attribute typeEmitter shall not be defined.

⌋*(RS_MANI_00039)*

[TPS_MANI_01212]{DRAFT} **Usage of attribute typeEmitter in the context of a CustomCppImplementationDataType** ⌈ Attribute typeEmitter does not have to be used in the context of a CustomCppImplementationDataType. If the typeEmitter is used regardless then the value of the attribute shall be set to the name of the header file that contains the language binding of the respective CustomCppImplementationDataType. ⌋*(RS_MANI_00039)*

[TPS_MANI_01169]{DRAFT} **Support for template data types** ⌈ Meta-class CppImplementationDataType supports the usage of templates for the definition of data types in C++ programs by means of the reference CppImplementationDataType.templateArgument.

The order of arguments in templates is significant, therefore templateArgument is modeled as an **ordered** collection. ⌋*(RS_MANI_00039)*

[TPS_MANI_01174]{DRAFT} **Semantics of reference in the role CppTemplateArgument.templateType** ⌈ Attribute CppTemplateArgument.templateType specifies the data type to be filled in the respective position of the template in the language binding. ⌋*(RS_MANI_00039)*

[TPS_MANI_01175]{DRAFT} **Semantics of reference in the role CppTemplateArgument.allocator** ⌈ Attribute CppTemplateArgument.allocator specifies the behavior of an allocator class to be filled in the respective position of the template in the language binding. ⌋*(RS_MANI_00039)*

[constr_1576]{DRAFT} **Existence of CppTemplateArgument.templateType vs. CppTemplateArgument.allocator** ⌈ For any given CppTemplateArgument, **at most one of** the references

- CppTemplateArgument.templateType or
- CppTemplateArgument.allocator

may exist. ⌋*()*

[TPS_MANI_01201]{DRAFT} **Standardized values for attribute CppTemplateArgument.category** ⌈ AUTOSAR reserves the following values for attribute CppTemplateArgument.category:

ASSOC_MAP_KEY : the specific CppTemplateArgument represents the *key* data type of an associative map.

ASSOC_MAP_VALUE : the specific CppTemplateArgument represents the *value* data type of an associative map.

⌋*(RS_MANI_00039)*

Class	**CppTemplateArgument**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class has the ability to define properties for template arguments. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
category	CategoryString	0..1	attr	This attribute shall be used to contribute further clarification regarding the semantics of the enclosing CppTemplateArgument.

▽

△

Class	CppTemplateArgument			
allocator	Allocator	0..1	ref	This reference identifies the applicable allocator. **Tags:** atp.Status=draft
inplace	Boolean	0..1	attr	This attribute specifies whether the shortName of the referenced templateType is used in the code generation and the type declaration is defined outside of the enclosing CppImplementationDataType (true) or whether the type definition is embedded inside of the enclosing CppImplementationDataType and the shortName is ignored (false).
templateType	CppImplementation DataType	0..1	ref	This reference identifies the data type of the specific template argument required for the language binding. **Tags:** atp.Status=draft

Table 3.15: CppTemplateArgument

[TPS_MANI_01171]{DRAFT} **Modeling of structured data types** ⌈ Meta-class `CppImplementationDataType` supports the creation of nested data types by means of the aggregation of `CppImplementationDataTypeElement` in the role `subElement`.

Because the order of sub-elements in a structured data type is significant the aggregation `subElement` is modeled as an **ordered** collection. ⌋*(RS_MANI_00039)*

Please note that although the modeling of structures is formally done using `CppImplementationDataType` it is actually only possible to use `StdCppImplementationDataType` for this purpose (see [constr_1578]).

Class	CppImplementationDataTypeElement			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. A CppImplementationDataTypeElement is used to represent an element of a structure, defining its type. **Tags:** atp.Status=draft			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing CppImplementationDataTypeElement as optional. This means the that, at runtime, the Cpp ImplementationDataTypeElement may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end.

▽

△

Class	CppImplementationDataTypeElement			
typeReference	CppImplementation DataTypeElement Qualifier	0..1	aggr	This aggregation defines the type of the Cpp ImplementationDataTypeElement and determines whether in C++ the CppImplementationDataTypeElement is defined inside or outside of the enclosing Cpp ImplementationDataType. **Tags:** atp.Status=draft

Table 3.16: CppImplementationDataTypeElement

Please note that there is no intention to support a “mixed” modeling of structured data types such that the resulting data type on C++ level would be composed of data types that are native to C++ and data types from the C subsystem.

While this would technically be possible on code level it would impose a huge effort on modeling level and the general consensus is that there is no real use case for such a "mixed" data type.

The C++ data type system can, as far as the implementation of the *AUTOSAR adaptive platform* is concerned, fully replace the “legacy” C data types in C++.

[constr_1572]{DRAFT} **Usage of `SwDataDefProps.implementationDataType` within a `CppImplementationDataType`** ⌈ Within the scope of a `CppImplementationDataType` the reference `CppImplementationDataType.swDataDefProps.implementationDataType` **shall not exist.** ⌋*()*

This aspect is also expressed in a more general form by [constr_1579].

As a consequence of [constr_1572], type references have to be done differently on the *AUTOSAR adaptive platform*. For this purpose dedicated references are available.

[TPS_MANI_01172]{DRAFT} **Description of type references in the scope of `CppImplementationDataType`** ⌈ The reference `CppImplementationDataType.typeReference` can be used to create a type reference from the enclosing `CppImplementationDataType` to another `CppImplementationDataType`. ⌋*(RS_MANI_00039)*

[TPS_MANI_01173]{DRAFT} **Description of type references in the scope of `CppImplementationDataTypeElement`** ⌈ `CppImplementationDataTypeElement.typeReference` can be used to create a reference to the `CppImplementationDataType` that shall apply for the enclosing `CppImplementationDataTypeElement`. ⌋*(RS_MANI_00039)*

Please note that the `CppImplementationDataTypeElement.typeReference` is realized as an Association Class that allows to add the `inplace` attribute to the `typeReference`.

Class	CppImplementationDataTypeElementQualifier			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This element qualifies the typeReference of the CppImplementationDataTypeElement to the Cpp ImplementationDataType. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
inplace	Boolean	0..1	attr	This attribute defines whether the member type of the CppImplementationDataTypeElement in C++ is an embedded type element inside of the enclosing struct (true) or whether the type declaration is defined outside of the struct.
typeReference	CppImplementation DataType	1	ref	This reference defines a type reference. **Tags:** atp.Status=draft

Table 3.17: CppImplementationDataTypeElementQualifier

[TPS_MANI_03196]{DRAFT} **Semantics of CppImplementationDataTypeElementQualifier.inplace attribute** ⌈ The CppImplementationDataTypeElementQualifier.inplace attribute defines whether the data type of the CppImplementationDataTypeElement in the C++ language binding is derived from the name or the properties of the referenced CppImplementationDataType.

Specifically, the following rules shall apply:

- if CppImplementationDataTypeElement.typeReference.inplace is set to False then the **shortName** of the CppImplementationDataType referenced in the role CppImplementationDataTypeElement.typeReference.typeReference shall be used in the C++ language binding.
- if CppImplementationDataTypeElement.typeReference.inplace is set to True then only the **properties** of the CppImplementationDataType referenced in the role CppImplementationDataTypeElement.typeReference.typeReference shall be used in the C++ language binding and the shortName is ignored.

⌋*(RS_MANI_00039)*

Please note that Figure 3.13 shows an example of a Structure where the typeReference of one subElement is classified as inplace.

[constr_1659]{DRAFT} **Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace** ⌈ The attribute CppImplementationDataTypeElementQualifier.inplace shall only exist if the target referenced in the role CppImplementationDataTypeElementQualifier.typeReference is an StdCppImplementationDataType that has attribute category set to either of the values

- ARRAY
- VECTOR

- ASSOCIATIVE_MAP
- VARIANT
- STRUCTURE
- STRING

⌋()

Rationale for the existence of [constr_1659]: by application of the exclusion principle, there are three cases where attribute CppImplementationDataTypeElementQualifier.inplace shall not exist:

- StdCppImplementationDataType of category VALUE
- CustomCppImplementationDataType
- CppImplementationDataType of category TYPE_REFERENCE

Neither of them can be used as a target of CppImplementationDataTypeElementQualifier.typeReference where CppImplementationDataTypeElementQualifier.inplace is set to False because in these cases there is already a valid name that is directly usable for the language binding and a possible indirection via a using clause would obviously require an additional name that is not available from the model.

After all, the motivation for the definition of a TYPE_REFERENCE is the direct opposite of the motivation behind using the attribute CppImplementationDataTypeElementQualifier.inplace to control the language binding. Therefore this case is also excluded.

[TPS_MANI_03201]{DRAFT} **Semantics of CppTemplateArgument.inplace attribute** ⌈ The CppTemplateArgument.inplace attribute defines whether the data type that is referenced by the templateType in the C++ language binding is derived from the name or the properties of the referenced CppImplementationDataType.

Specifically, the following rules shall apply:

- if CppTemplateArgument.inplace is set to False then the **shortName** of the CppImplementationDataType referenced in the role CppTemplateArgument.templateType shall be used in the C++ language binding.
- if CppTemplateArgument.inplace is set to True then only the **properties** of the CppImplementationDataType referenced in the role CppTemplateArgument.templateType shall be used in the C++ language binding and the shortName is ignored.

⌋*(RS_MANI_00039)*

[constr_1660]{DRAFT} **Restriction for the usage of CppTemplateArgument.inplace** ⌈ The attribute CppTemplateArgument.inplace shall only exist if the target referenced in the role CppTemplateArgument.templateType is an StdCppImplementationDataType that has attribute category set to either of the values

- ARRAY
- VECTOR
- ASSOCIATIVE_MAP
- VARIANT
- STRUCTURE
- STRING

⌋*()*

Rationale for the existence of [constr_1660]: by application of the exclusion principle, there are three cases where attribute CppTemplateArgument.inplace shall not exist:

- StdCppImplementationDataType of category VALUE
- CustomCppImplementationDataType
- CppImplementationDataType of category TYPE_REFERENCE

Neither of them can be used as a target of CppTemplateArgument.templateType where CppTemplateArgument.inplace is set to False because in these cases there is already a valid name that is directly usable for the language binding and a possible indirection via a using clause would obviously require an additional name that is not available from the model.

After all, the motivation for the definition of a TYPE_REFERENCE is the direct opposite of the motivation behind using the attribute CppTemplateArgument.inplace to control the language binding. Therefore this case is also excluded.

Please note that the question of the value of attribute CppTemplateArgument.inplace for the case of CppTemplateArgument.templateType referring to StdCppImplementationDataType of category STRUCTURE is regulated by [constr_3462].

[constr_3462]{DRAFT} **CppTemplateArgument.templateType reference to StdCppImplementationDataType of category STRUCTURE and the inplace flag** ⌈ CppTemplateArgument.templateType that points to a StdCppImplementationDataType of category STRUCTURE shall have the inplace attribute set to **false**. ⌋*()*

The reason for [constr_3462] is that the usage of an unnamed struct as template argument is not permitted by ISO C++11/14/17.

[constr_3446]{DRAFT} **CppTemplateArgument with allocator reference and the inplace flag** ⌈ A CppTemplateArgument that points with an allocator reference to an Allocator shall not have the inplace flag set to a value. ⌋*()*

Class	Allocator			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the ability to take influence on the way objects are allocated in memory, for example it can be controlled whether an objects is allocated on the heap or on the stack. **Tags:** atp.Status=draft atp.recommendedPackage=Allocators			
Base	*ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition of a namespace of an Allocator. **Tags:** atp.Status=draft

Table 3.18: Allocator

[TPS_MANI_01100]{DRAFT} **Semantics of Allocator** ⌈ Meta-class Allocator carries the ability to define the properties of an allocation of memory. The general approach for memory allocation is expressed by means of the attribute category.

The following values of Allocator.category are standardized by AUTOSAR:

- MAX_SIZE_HEAP: when using this allocator there is the intention to allocate a fixed-size chunk on the heap. This allocator adds the ability to define a maximum number of elements to the semantics of the default allocator of ara::core::Vector.
- MAX_SIZE_STACK: when using this allocator there is the intention to allocate a fixed-size chunk on the stack. Memory on the stack always needs to be constrained in terms of the maximum size. In other words, there is hardly any case where an unbounded amount of memory should be allocated on the stack.
- MAX_SIZE_DATASEGMENT: when using this allocator there is the intention to allocate a fixed-size chunk in the data segment.

⌋*(RS_MANI_00016)*

[constr_1578]{DRAFT} **applicable data categories** ⌈ Table 3.19 defines the applicable categorys vs. meta-class. ⌋*()*

Category	Applicable to ...								Description
	ApplicationArrayDataType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValueSpecification	StdCppImplementationDataType	CustomCppImplementationDataType	
VALUE			x	x	x	x	x		Contains a single value. See also [TPS_MANI_03192].
TYPE_REFERENCE							x		The element is defined via reference to another data type (via CppImplementationDataType.typeReference.
STRUCTURE		x		x	x		x		Holds one or several further elements which can have different AutosarDataTypes. See also [TPS_MANI_03180].
VARIANT							x	x	Can hold values of different data types. It is similar to STRUCTURE except that all of its members start at the same location in memory. A VARIANT data prototype can contain only one of its elements at a time and represents a type-safe union. The size of the VARIANT is at least the size of the largest member. See also [TPS_MANI_03189].
ARRAY	x			x	x		x	x	A fixed-sized array of sub-elements of the same type. See also [TPS_MANI_03169].
VECTOR							x	x	An array of elements of the same time that is able to grow at run-time. See also [TPS_MANI_03174].
ASSOCIATIVE_MAP							x	x	An associative array of key-value pairs. See also [TPS_MANI_03183].
STRING			x	x	x	x	x		Contains a text string. See also [TPS_MANI_03178].
BOOLEAN			x	x	x	x			Contains one boolean state. Depending on the CPU direct addressing of single bits may not be available. So a byte or a word can be used to store only one logical state.

Table 3.19: Usage of category for Data Types

3.3.3.2 Attributes of SwDataDefProps

[constr_1579]{DRAFT} **SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*** ⌈ A complete list of the SwDataDefProps and other attributes and their multiplicities which are allowed for a given category is shown in table 3.20. ⌋*()*

A consequence of [constr_1578] is that the Table 3.20 shows only the values of category that are limited to the *AUTOSAR adaptive platform*. For all other values of category that are also supported on the *AUTOSAR classic platform* please refer to a similar table contained in the specification of the Software Component Template [1].

Attributes of SwDataDefProps	Root Element	Attribute Existence per Category							
	CppImplementationDataType	VALUE	TYPE_REFERENCE	STRUCTURE	VARIANT	ARRAY	VECTOR	ASSOCIATIVE_MAP	STRING
additionalNativeTypeQualifier									
annotation	x	*	*	*	*	*	*	*	*
baseType									
compuMethod	x	0..1	0..1						
dataConstr.dataConstrRule.physConstrs	x	d/c[4]	d/c			d/c	d/c		
dataConstr.dataConstrRule.internalConstrs	x	0..1	0..1			0..1	0..1		
displayFormat	x	0..1		0..1	0..1	0..1	0..1	0..1	0..1
implementationDataType									
invalidValue	x	0..1	0..1	0..1		0..1			
stepSize									
swAddrMethod									
swAlignment									
swBitRepresentation									
swCalibrationAccess									
swCalprmAxisSet									
swComparisonVariable									
swDataDependency									
swHostVariable									
swImplPolicy									
swIntendedResolution									
swInterpolationMethod									
swIsVirtual									
swPointerTargetProps									
swPointerTargetProps.swDataDefProps									
swPointerTargetProps.functionPointerSignature									
swRecordLayout									
swRefreshTiming	x	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swTextProps									
swValueBlockSize									
unit									
valueAxisDataType									

▽

[4] don't care

△

Attributes of SwDataDefProps	Root Element	Attribute Existence per Category							
	CppImplementationDataType	VALUE	TYPE_REFERENCE	STRUCTURE	VARIANT	ARRAY	VECTOR	ASSOCIATIVE_MAP	STRING
Other Attributes									
subElement: CppImplementationDataTypeElement	x			1..*					
templateArgument	x				1..*	1	1..*	2..*	0..1
typeReference	x		1						

Table 3.20: Allowed Attributes vs. category for CppImplementationDataType

3.3.3.3 Primitive Data Types

[TPS_MANI_03192]{DRAFT} **CppImplementationDataType of category VALUE** ⌈ The primitive data types like Boolean, fixed-width integer data types and floating-point data types are described as CppImplementationDataTypes of category VALUE. ⌋*(RS_MANI_00039)*

[TPS_MANI_03193]{DRAFT} **CppImplementationDataType of category TYPE_REFERENCE** ⌈ The definition of a CppImplementationDataType of category TYPE_REFERENCE creates an alias for another CppImplementationDataType that is referenced by the typeReference. ⌋*(RS_MANI_00039)*

3.3.3.4 String Data Type

[TPS_MANI_03178]{DRAFT} **StdCppImplementationDataType of category STRING** ⌈ A StdCppImplementationDataType of category STRING represents a container data type for a sequence of characters.

AUTOSAR demands that the C++ binding of a StdCppImplementationDataType of category STRING is implemented by a ara::core::String. ⌋*(RS_MANI_00039)*

[constr_1674]{DRAFT} **Suported encoding of StdCppImplementationDataType of category STRING** ⌈ On the level of the meta-model (and, by extension, the language binding), the only supported encoding of StdCppImplementationDataType of category STRING is UTF-8. ⌋()

Please note that it is nonetheless possible to use a different encoding, e.g. UTF-16 on the level of a SOME/IP message. This behavior can be configured by means of ApSomeipTransformationProps. As a consequence, a transcoding may have to be applied between the representation of a string on the wire and in the software.

[TPS_MANI_03179]{DRAFT} **C++ language binding of StdCppImplementationDataTypes of category STRING** ⌈ A CppImplementationDataType of category STRING shall be implemented as ara::core::String. ⌋*(RS_MANI_00039)*

The formulation of [TPS_MANI_03179] leaves room for potential later extensions towards the support for other storage formats.

The example depicted in Figure 3.8 contains the definition of both an ApplicationDataType as well as the definition of the corresponding CppImplementationDataType.

The latter obviously becomes significantly lighter to model thanks to the restriction that, as far as the C++ language binding is concerned, a CppImplementationDataType of category STRING shall only be implemented on the basis of an ara::core::String.

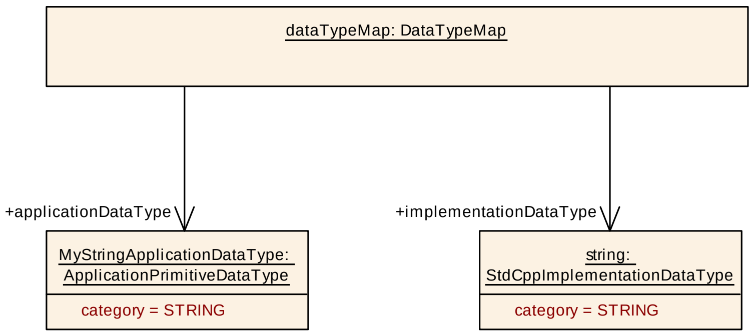


Figure 3.8: Example of the model of a string with UTF-8 encoding

Another aspect of the example in Figure 3.8 is that it defines the intended encoding of the modeled data type in the scope of the ApplicationPrimitiveDataType.

[TPS_MANI_03188]{DRAFT} **Usage of an Allocator for a StdCppImplementationDataType of category STRING** ⌈ A StdCppImplementationDataType of category STRING is allowed to aggregate a CppTemplateArgument that refers to an Allocator with the allocator reference. ⌋*(RS_MANI_00039)*

3.3.3.5 Array Data Type

[TPS_MANI_03169]{DRAFT} **CppImplementationDataType with fixed size array semantics** ⌈ A CppImplementationDataType of category ARRAY represents a container data type that encapsulates fixed size arrays. ⌋*(RS_MANI_00039)*

[TPS_MANI_03170]{DRAFT} **CppImplementationDataType of category ARRAY** ⌈ For a C++ binding, a CppImplementationDataType of category ARRAY can be implemented as

- an ara::core::Array if StdCppImplementationDataType subclass is used for modeling or as
- an array type in a custom namespace (e.g. my::array) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

⌋*(RS_MANI_00039)*

[TPS_MANI_03171]{DRAFT} **Value type of a CppImplementationDataType of category ARRAY** ⌈ The type of elements contained in a CppImplementationDataType of category ARRAY is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument. ⌋*(RS_MANI_00039)*

[constr_3433]{DRAFT} **Aggregation of templateArguments for an ARRAY** ⌈ CppImplementationDataType of category ARRAY that boils down to ara::core::Array shall aggregate exactly one templateArgument that defines the type of elements contained in the CppImplementationDataType of category ARRAY. ⌋*()*

[TPS_MANI_03172]{DRAFT} **Size of a CppImplementationDataType of category ARRAY** ⌈ The primitive attribute arraySize of a CppImplementationDataType of category ARRAY shall be used to define the size of the array. ⌋*(RS_MANI_00039)*

Figure 3.9 shows an example of an one-dimensional array of uint16 elements with arraySize = 5.

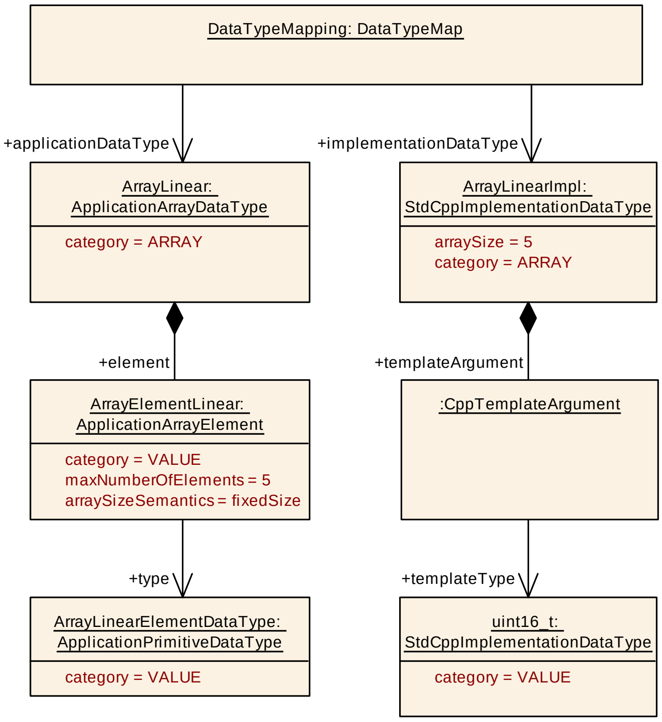


Figure 3.9: Example of the model of a one-dimensional array

[TPS_MANI_03173]{DRAFT} **Definition of a multidimensional Array** ⌈ A multidimensional CppImplementationDataType of category ARRAY contains nested CppImplementationDataTypes of category ARRAY.

The CppImplementationDataType of category ARRAY that represents the outer array will refer to a CppImplementationDataType of category ARRAY that represents the inner array via the aggregated templateArgument. Such a definition describes a two-dimensional Array; consequently a type with more dimensions is described by just nesting more CppImplementationDataTypes of category ARRAY.

The array element itself is specified by the innermost CppImplementationDataType with category different from ARRAY. ⌋*(RS_MANI_00039)*

Figure 3.10 shows an example of a multidimensional array where a CppImplementationDataType of category ARRAY with arraySize = 5 has a templateArgument that points to the inner CppImplementationDataType of category ARRAY in the role templateType.

The inner CppImplementationDataType has a templateArgument that finally points with the templateType reference to a primitive type.

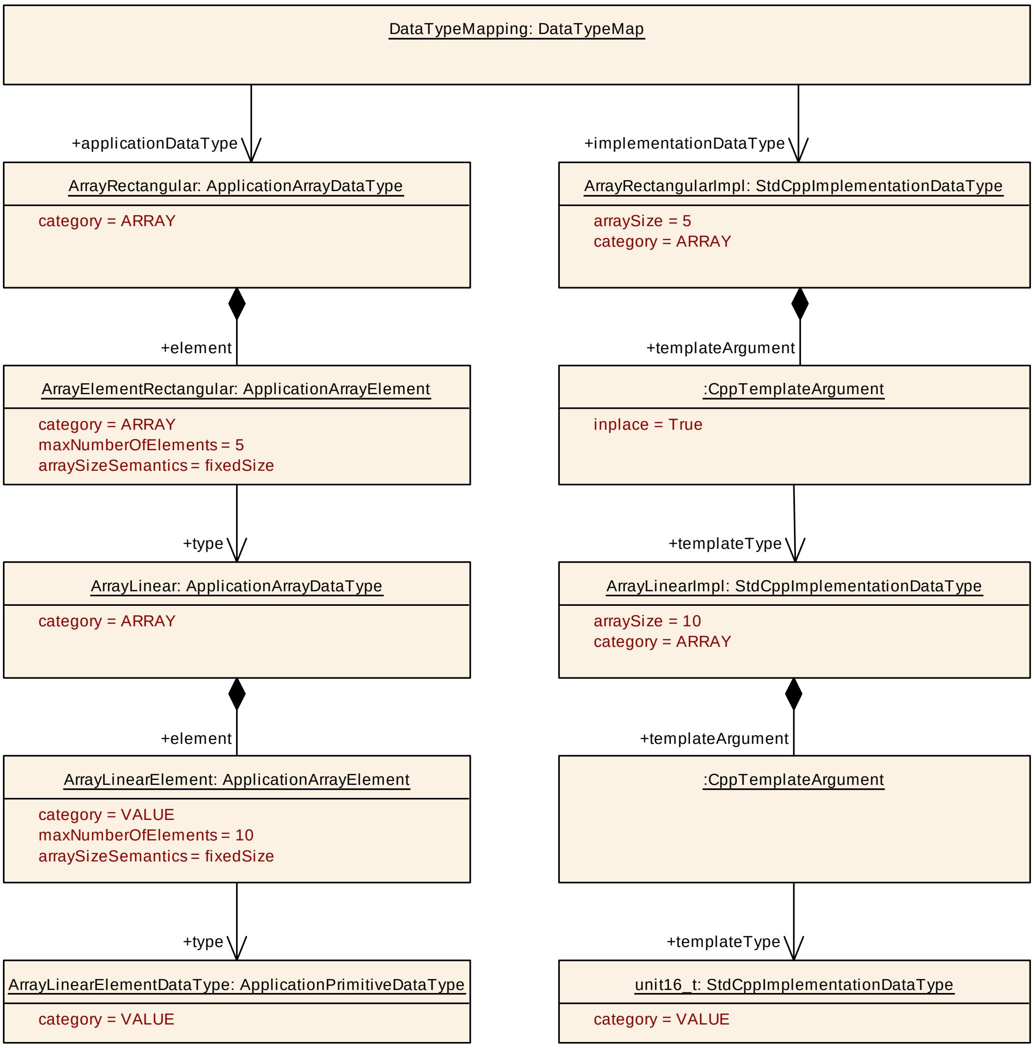


Figure 3.10: Example of the model of a multidimensional array

Such a model will result in the following C++ code since the `CppTemplateArgument.inplace` flag is set to true for the outer array:

```
using ArrayRectangularImpl = ara::core::Array<ara::core::Array<uint16_t,
  10>, 5>;
```

3.3.3.6 Vector Data Type

[TPS_MANI_03174]{DRAFT} **`CppImplementationDataType` with variable size array semantics** ⌈ A `CppImplementationDataType` of `category VECTOR` represents a container data type that encapsulates variable size arrays. ⌋*(RS_MANI_00039)*

[TPS_MANI_03175]{DRAFT} **`CppImplementationDataType` of `category VECTOR`** ⌈ For a C++ binding, a `CppImplementationDataType` of `category VECTOR` can be implemented as

- an ara::core::Vector if StdCppImplementationDataType subclass is used or as
- a vector type in a custom namespace (e.g. my::vector) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

⌋*(RS_MANI_00039)*

[TPS_MANI_03176]{DRAFT} **Value type of a CppImplementationDataType of category VECTOR** ⌈ The type of elements contained in a CppImplementationDataType of category VECTOR is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument. ⌋*(RS_MANI_00039)*C

[constr_3434]{DRAFT} **Aggregation of templateArguments for a VECTOR** ⌈ CppImplementationDataType of category VECTOR that boils down to ara::core::Vector shall aggregate

- one templateArgument that defines the type of elements contained in the CppImplementationDataType of category VECTOR with the templateType reference.
- optionally one additional templateArgument that defines the Allocator with the allocator reference.

⌋*()*

[TPS_MANI_03186]{DRAFT} **Usage of arraySize in case of a Vector** ⌈ If the CppImplementationDataType of category VECTOR aggregates a templateArgument that defines the Allocator with the allocator reference then the attribute arraySize that defines the maximum size of the vector is allowed to be used. ⌋*(RS_MANI_00039)*

Figure 3.11 shows an example of an one-dimensional vector of uint16 elements.

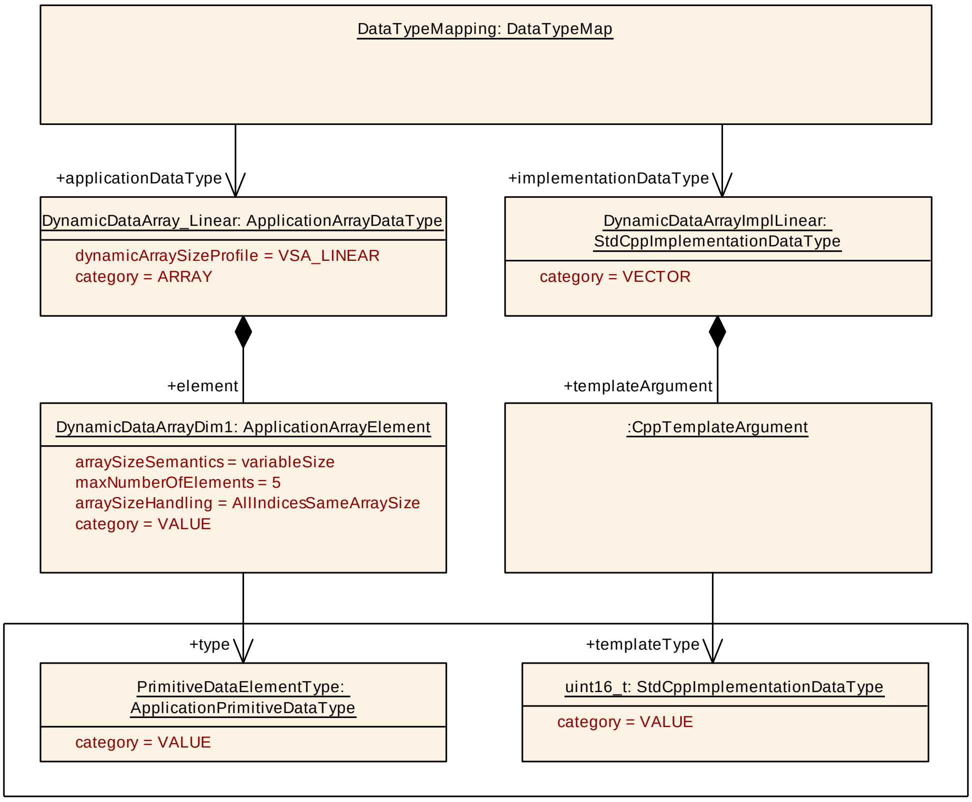


Figure 3.11: Example of the model of a one-dimensional vector

[TPS_MANI_03177]{DRAFT} **Definition of a multidimensional Vector** ⌈ A multidimensional CppImplementationDataType of category VECTOR contains nested CppImplementationDataTypes of category VECTOR.

The CppImplementationDataType of category VECTOR that represents the outer vector will refer to a CppImplementationDataType of category VECTOR that represents the inner vector via the aggregated templateArgument.

Such a definition describes a two-dimensional Vector; consequently a type with more dimensions is described by just nesting more CppImplementationDataTypes of category VECTOR.

The vector element itself is specified by the innermost CppImplementationDataType with category different from VECTOR. ⌋*(RS_MANI_00039)*

Figure 3.12 shows an example of a multidimensional vector where a CppImplementationDataType of category VECTOR has a templateArgument that points to the inner CppImplementationDataType of category VECTOR in the role templateType. The inner CppImplementationDataType has a templateArgument that finally points with the templateType reference to a primitive type.

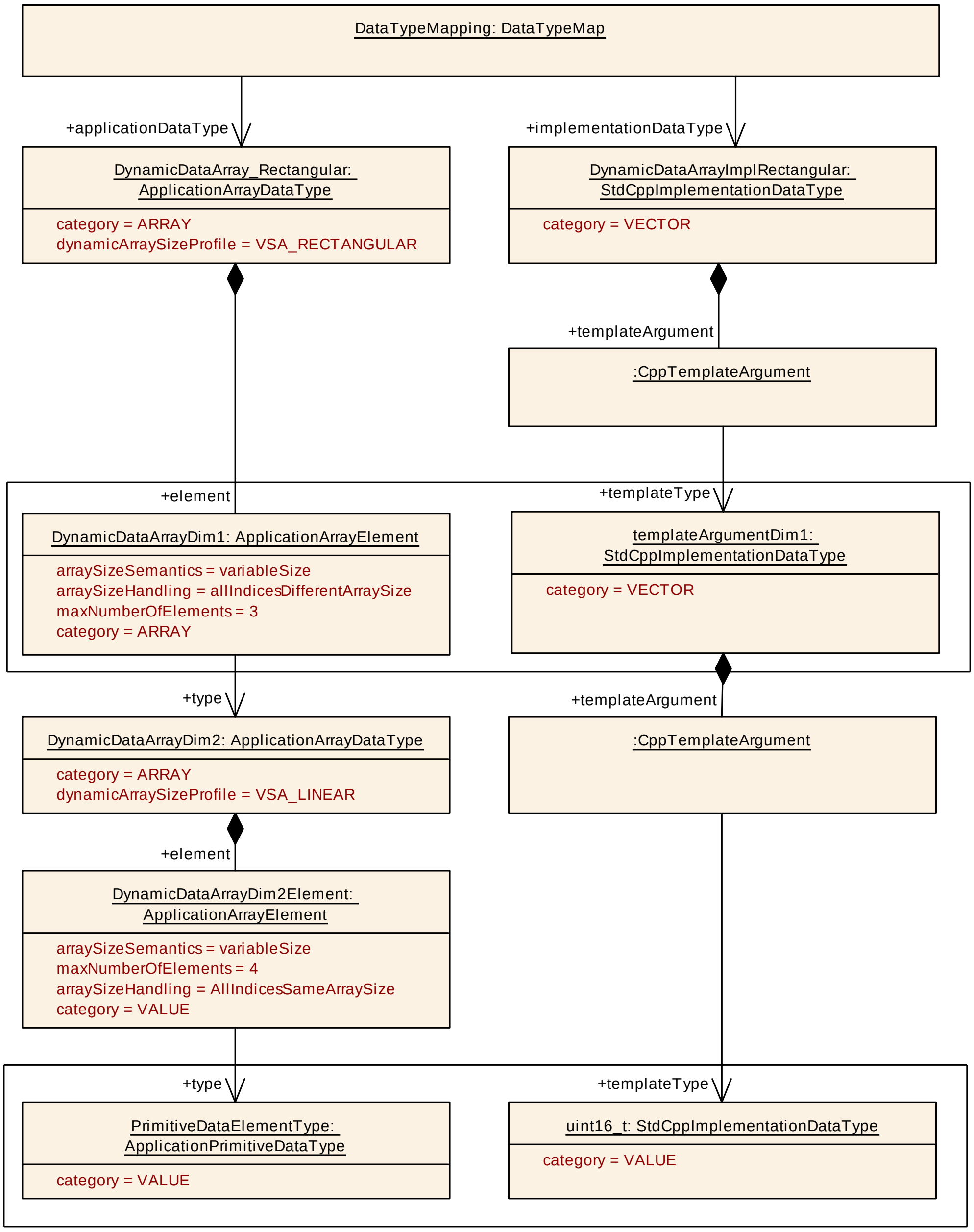


Figure 3.12: Example of the model of a multidimensional vector

Such a model will result in the following C++ code since the CppTemplateArgument.inplace flag is not set for the inner vector:

```
using templateArgumentDim1 = ara::core::Vector<uint16_t>;
using DynamicDataArrayImplRectangular = ara::core::Vector<
    templateArgumentDim1>;
```

Please note that the meta-model supports the creation of a reference to a specific element (identified by means of the index) of a CppImplementationDataType of category VECTOR.

However, this may lead to a problem at run-time if the specific element does not exist at the respective point in time. Any software using such data types needs to be prepared for the potential non-existence of vector elements.

Alternatively, it could be an option to simply avoid a situation where an element of a CppImplementationDataType of category VECTOR becomes the target of a reference in the model.

3.3.3.7 Struct Data Type

[TPS_MANI_03180]{DRAFT} **Definition of Structures** ⌈ A StdCppImplementationDataType of category STRUCTURE represents a data type for holding an ordered collection of variables of arbitrary data types. ⌋*(RS_MANI_00039)*

[TPS_MANI_03181]{DRAFT} **Definition of members in StdCppImplementationDataType of category STRUCTURE** ⌈ Members in a StdCppImplementationDataType of category STRUCTURE are defined by ordered CppImplementationDataTypeElements that are aggregated in the role subElement by the enclosing StdCppImplementationDataType of category STRUCTURE.

The name of each member is defined by the shortName of the CppImplementationDataTypeElement.

The type of each member is defined by the typeReference to a CppImplementationDataType. ⌋*(RS_MANI_00039)*

Please note that the inplace flag that is able to classify a CppImplementationDataTypeElement.typeReference is documented in [TPS_MANI_03196].

The example depicted in Figure 3.13 shows the definition of a Structure, called MyStruct, that has two members. The typeReference of the subElements with the shortName ArrayElement is classified with inplace = **True** and the following struct declaration is generated out of the model:

```
struct MyStruct {
    ara::core::uint8_t PrimitiveElement;
    ara::core::Array<uint8_t,5> ArrayElement;
};
```

In case that the inplace attribute in the typeReference to the array is set to **False** the model results in a using-declaration of ArrayDataType that is defined outside of MyStruct.

```
using ArrayDataType = ara::core::Array<uint8_t,5>;

struct MyStruct {
    std::uint8_t PrimitiveElement;
    ArrayDataType ArrayElement;
};
```

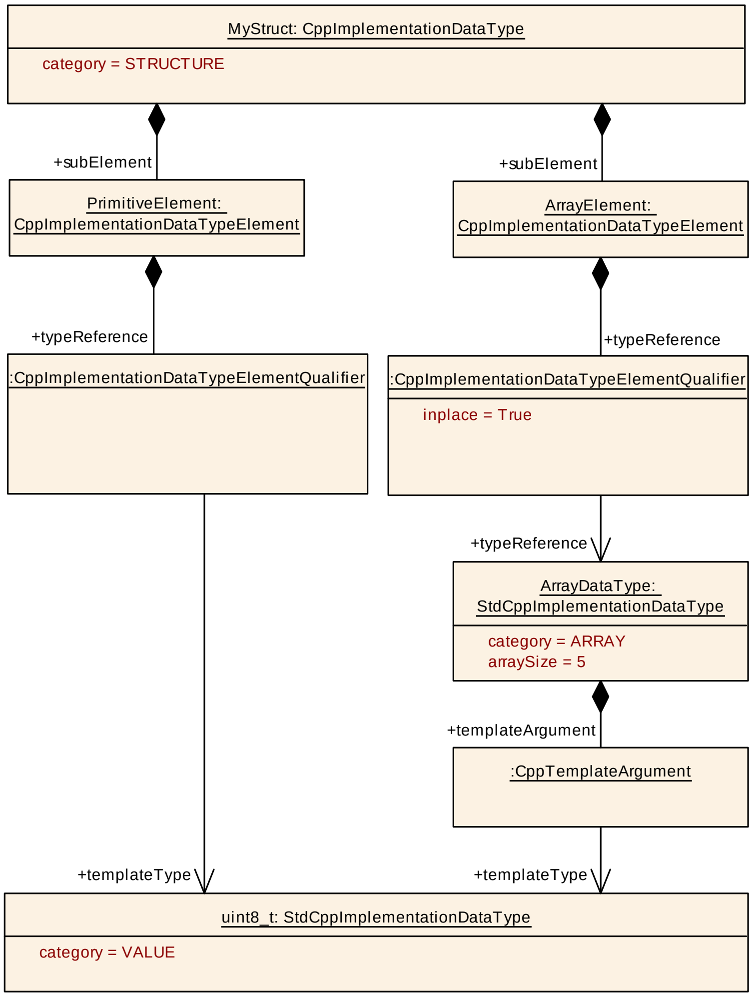


Figure 3.13: Example of the model of a Struct

3.3.3.8 Enumeration Data Type

[TPS_MANI_03187]{DRAFT} **Definition of enumeration types** ⌈ In the AUTOSAR meta-model, an enumeration is not implemented by means of a CppImplementationDataType with an own category.

Instead, a discrete set of integer numbers can be used as a structural description for a single fundamental CppImplementationDataType of category TYPE_REFERENCE that boils down to a CppImplementationDataType of category VALUE.

The mapping of the integer numbers to labels in the scope of the definition of an enumeration is considered part of the semantical definition via an attached CompuMethod with category TEXTTABLE rather than part of the structural description. ⌋ *(RS_MANI_00039)*

The rules for the usage of a CompuMethod with category TEXTTABLE are the same as in the AUTOSAR Classic Platform and are described in the Software Component Template [1].

To summarize, an enumeration value in the CompuMethod with category TEXTTABLE can be provided as a text value in the vt of the CompuConst, in the shortLabel or symbol of the applicable CompuScale of the CompuMethod.

Each CompuScale shall be defined as compuInternalToPhys computation in the CompuMethod and shall contain an upperLimit and lowerLimit.

The following example illustrates how an enumeration is specified using a CompuMethod.

Listing 3.2: example for enumeration

```
<COMPU-METHOD>
  <SHORT-NAME>cylinders</SHORT-NAME>
  <CATEGORY>TEXTTABLE</CATEGORY>
  <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder1</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder2</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">2</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder3</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">3</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">3</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>Cylinder4</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
    </COMPU-SCALES>
  </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

3.3.3.9 Map Data Type

[TPS_MANI_03183]{DRAFT} **CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈ A CppImplementationDataType of category ASSOCIATIVE_MAP represents a container that contains key-value pairs with unique keys. ⌋ *(RS_MANI_00039)*

[TPS_MANI_03184]{DRAFT} **CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈ For a C++ binding, a CppImplementationDataType of category ASSOCIATIVE_MAP can be implemented as

- an ara::core::Map if StdCppImplementationDataType subclass is used or as
- a map type in a custom namespace (e.g. my::map) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

⌋ *(RS_MANI_00039)*

[TPS_MANI_03185]{DRAFT} **Structure of a CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈ A CppImplementationDataType of category ASSOCIATIVE_MAP that boils down to a ara::core::Map shall aggregate the following CppTemplateArguments:

- the **first** CppTemplateArgument shall refer to a CppImplementationDataType with the templateType reference.

  This CppTemplateArgument represents the role that corresponds to ApplicationAssocMapDataType.key and defines the respective data type details.
- the **second** CppTemplateArgument shall refer a CppImplementationDataType with the templateType reference.

  This CppTemplateArgument represents the role that corresponds to ApplicationAssocMapDataType.value and defines the respective data type details.
- the optional **third** CppTemplateArgument shall refer to an Allocator with the allocator reference.

⌋ *(RS_MANI_00039)*

The example depicted in Figure 3.14 shows the definition of a ASSOCIATIVE_MAP that has two CppTemplateArguments, one for the key and one for the value.

Please note that the CppTemplateArguments of a CppImplementationDataType are ordered in ARXML and this order is not visible in the object diagram.

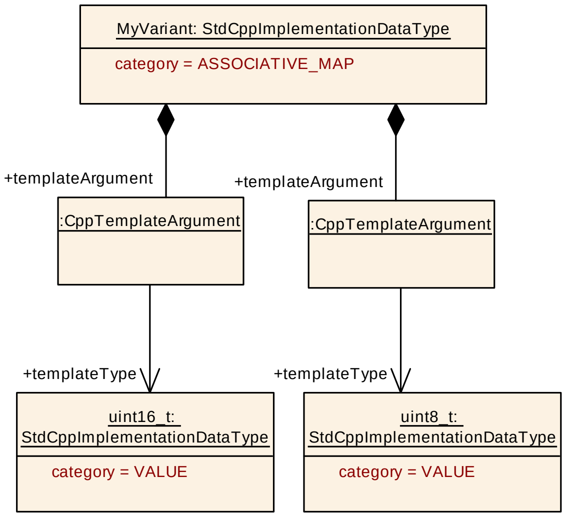


Figure 3.14: Example of the model of an ASSOCIATIVE_MAP

3.3.3.10 Variant Data Type

[TPS_MANI_03189]{DRAFT} **Definition of CppImplementationDataType of category VARIANT** ⌈ A CppImplementationDataType of category VARIANT represents a type safe union. ⌋*(RS_MANI_00039)*

[TPS_MANI_03190]{DRAFT} **CppImplementationDataType of category VARIANT** ⌈ For a C++ binding, a CppImplementationDataType of category VARIANT can be implemented as

- an ara::core::Variant if StdCppImplementationDataType subclass is used or as
- a variant type in a custom namespace (e.g. my::variant) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

⌋*(RS_MANI_00039)*

[TPS_MANI_03191]{DRAFT} **Definition of type alternatives stored in a VARIANT** ⌈ A type alternative that is stored in a CppImplementationDataType of category VARIANT is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument. ⌋*(RS_MANI_00039)*

[constr_3429]{DRAFT} **No allocator usage for CppImplementationDataTypes of category VARIANT** ⌈ CppImplementationDataType of category VARIANT is not allowed to aggregate a templateArgument that points to an Allocator in the role allocator. ⌋*()*

The example depicted in Figure 3.15 shows the definition of a VARIANT that has two CppTemplateArguments. Each one represents one alternative type. Please note that the CppTemplateArguments of a CppImplementationDataType are ordered in ARXML and this order is not visible in the object diagram.

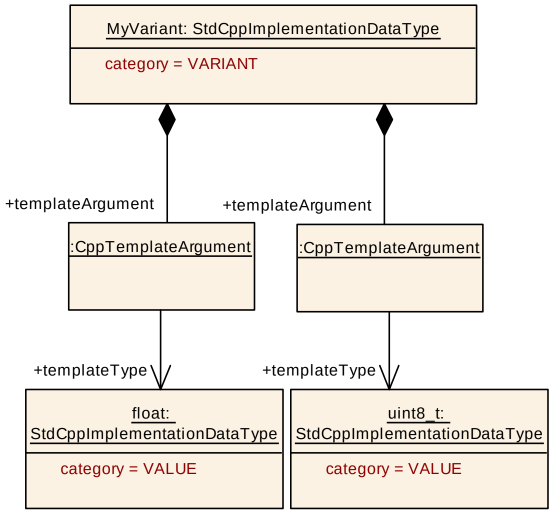


Figure 3.15: Example of the model of an VARIANT

3.3.3.11 Bitfield Data Type

[TPS_MANI_03202]{DRAFT} **Definition of bitfield types** ⌈ In the AUTOSAR meta-model, a bitfield is not implemented by means of a CppImplementationDataType with an own category.

A bitfield is defined in the context of a primitive StdCppImplementationDataType of category TYPE_REFERENCE that boils down to a StdCppImplementationDataType of category VALUE.

A CompuMethod of category BITFIELD_TEXTTABLE is used to assign a special meaning to each bit of the primitive StdCppImplementationDataType. ⌋ *(RS_MANI_00039)*

CompuScales with a mask inside of the CompuMethod of category BITFIELD_TEXTTABLE are defining isolated parts that can be independent from each other with respect to the semantics of the data that match the mask.

The rules for the usage of a CompuMethod with category BITFIELD_TEXTTABLE are the same as in the AUTOSAR Classic Platform and are described in the Software Component Template [1].

3.3.4 Compatibility of ApplicationDataType and CppImplementationDataType

The usage of ApplicationDataTypes implies that also a corresponding CppImplementationDataType exists at a certain point in time. The usage of CppImplementationDataTypes in a ServiceInterface is required as the basis for generating the ara::com proxies and skeletons and as basis for the serialization of the payload in the network binding.

[TPS_MANI_03223]{DRAFT} **Existence of CppImplementationDataType** ⌈ The existence of CppImplementationDataTypes is **not** required until the methodology step of generating the Service header files for a ServiceInterface. Before arriving at this step in the methodology, it is perfectly feasible to use only ApplicationDataTypes for describing the semantics of ServiceInterfaces. ⌋ *(RS_MANI_00003)*

As a consequence, it is necessary to define compatibility rules that unambiguously clarify the conformance of an ApplicationDataType with a CppImplementationDataType and vice versa.

Several rules depend on the category of the data types:

1. As a general rule, if a CppImplementationDataType of category TYPE_REFERENCE is targeted by a type mapping all the rules given below apply to the CppImplementationDataType which is finally valid after resolving all such references.

   This is not repeated in all rules. For example, if the document states that a given ApplicationDataType can be mapped to a CppImplementationDataType of category VALUE this shall include the possibility of mapping to a CppImplementationDataType of category TYPE_REFERENCE which refers to another CppImplementationDataType of category VALUE.

2. **[constr_5033]**{DRAFT} **Compatibility of data types with category VALUE** ⌈ An ApplicationDataType of category VALUE can only be mapped to a CppImplementationDataType which also has category VALUE. ⌋*()*

   In this case, the C++ data type resulting from the CppImplementationDataType shall be able to express all the numerical values required by the ApplicationDataType.

   This condition is fulfilled if the numerical range which can be expressed by the C++ data type at least covers the range defined by the limits in ApplicationDataType.swDataDefProps.dataConstr (which are either internal limits or physical limits to be converted via the CompuMethod which also has to be provided by the ApplicationDataType).

   The condition is also fulfilled if the C++ data type covers the range defined in the CompuMethod for an enumeration.

3. **[constr_5034]**{DRAFT} **Compatibility of data types with category BOOLEAN** ⌈ An ApplicationDataType of category BOOLEAN can only be mapped to a CppImplementationDataType of category VALUE. ⌋*()*

4. **[constr_5035]**{DRAFT} **Compatibility of data types with category STRING** ⌈ An ApplicationDataType of category STRING can only be mapped to a CppImplementationDataType of category STRING. ⌋*()*

5. **[constr_5036]**{DRAFT} **Compatibility of data types with category ARRAY** ⌈ An ApplicationDataType of category ARRAY can only be mapped to

   - a CppImplementationDataType of category ARRAY **or**
   - a CppImplementationDataType of category VECTOR.

   ⌋*()*

   In this case, the array size and the type of the array elements of the CppImplementationDataType shall be such that they can be mapped/transferred 1:1 by order to the corresponding application data and vice versa.

6. **[constr_5037]**{DRAFT} **Compatibility of data types with category ARRAY with variableSize** ⌈ An ApplicationDataType of category ARRAY that includes one ApplicationArrayElement with arraySizeSemantics set to variableSize in one of the defined dimensions shall be mapped to

   - a CppImplementationDataType of category VECTOR

   ⌋*()*

7. **[constr_5038]**{DRAFT} **Compatibility of data types with category ARRAY with fixedSize** ⌈ An ApplicationDataType of category ARRAY that includes only ApplicationArrayElements with arraySizeSemantics set to fixedSize in all defined dimensions shall be mapped to

   - a CppImplementationDataType of category ARRAY

   ⌋*()*

8. **[constr_5039]**{DRAFT} **Compatibility of data types with category STRUCTURE** ⌈ An ApplicationDataType of category STRUCTURE can only be mapped to a CppImplementationDataType of category STRUCTURE. ⌋*()*

   This means, that the corresponding pairs of elements shall also have compatible types.

9. **[constr_5040]**{DRAFT} **Compatibility of ApplicationRecordDataType and CppImplementationDataType that both represent an Optional Element Structure** ⌈ An ApplicationRecordDataType that represents an Optional Element Structure can only be mapped to a CppImplementationDataType of category STRUCTURE that represents an Optional Element Structure if corresponding pairs of elements have the same value of the attribute isOptional. ⌋*()*

10. **[constr_5041]**{DRAFT} **Compatibility of data types with category ASSOCIATIVE_MAP** ⌈ An ApplicationDataType of category ASSOCIATIVE_MAP can only be mapped to a CppImplementationDataType of category ASSOCIATIVE_MAP. ⌋*()*

11. **[constr_5042]**{DRAFT} **No data type mapping for CppImplementationDataType of category VARIANT** ⌈ An ApplicationDataType shall never be mapped to a CppImplementationDataType of category VARIANT. ⌋*()*

12. **[constr_5043]**{DRAFT} **Forbidden mappings to CppImplementationDataType** ⌈ An ApplicationDataType of category COM_AXIS, RES_AXIS, CURVE, MAP, CUBOID, CUBE_4, CUBE_5 is not supported by the Adaptive Platform and can therefore not be mapped to a CppImplementationDataType. ⌋*()*

Please note that the categories listed in [constr_5043] are not supported because there is no use case for the usage in Adaptive Platform.

On the AUTOSAR classic Platform, elements of a composite data type are not required to be considered in a DataTypeMap. This regulation is motivated by the fact that an element of a composite data type on the AUTOSAR classic Platform does not necessarily have a reference to an ImplementationDataType.

On the AUTOSAR adaptive Platform the situation is different. The CppImplementationDataTypeElement always requires a reference to a formalized CppImplementationDataType.

Since the processing of the data type definition becomes much easier if all the relevant data types are mentioned in a DataTypeMap the existence of [constr_5044] is motivated.

[constr_5044]{DRAFT} **DataTypeMap for composite data types** ⌈ In the context of a given ServiceInterface, all pairs of ApplicationDataType and CppImplementationDataType used in the context of the definition of an ApplicationCompositeDataType used in the context of an event, field, method shall be described in a DataTypeMap that is contained in one of the DataTypeMappingSets that are referenced in a PortInterfaceToDataTypeMapping that also references the mentioned ServiceInterface. ⌋*()*

3.4 Service Interface

3.4.1 Overview

[TPS_MANI_01001]{DRAFT} **Meaning of ServiceInterface** ⌈ Meta-class ServiceInterface inherits from PortInterface and allows for a heterogeneous aggregation of elements, i.e. it is possible to mix

- aggregation of VariableDataPrototype in the role event with

- aggregation of meta-class `Field` in the role `field` with
- aggregation of `ClientServerOperation` in the role `method`

within the same `ServiceInterface`. ⌋*(RS_MANI_00003)*

The purpose of this modeling is to embrace the concept of service-oriented communication [3] and better support this paradigm for communication on the *AUTOSAR adaptive platform*.

Please note that, in terms of semantics, the `ApApplicationError` represents sort of a second-class citizen (that only makes sense in the presence of `ClientServerOperation` in the role `method`) in the scope of the `ServiceInterface`.

More information can be found in section 3.4.7.

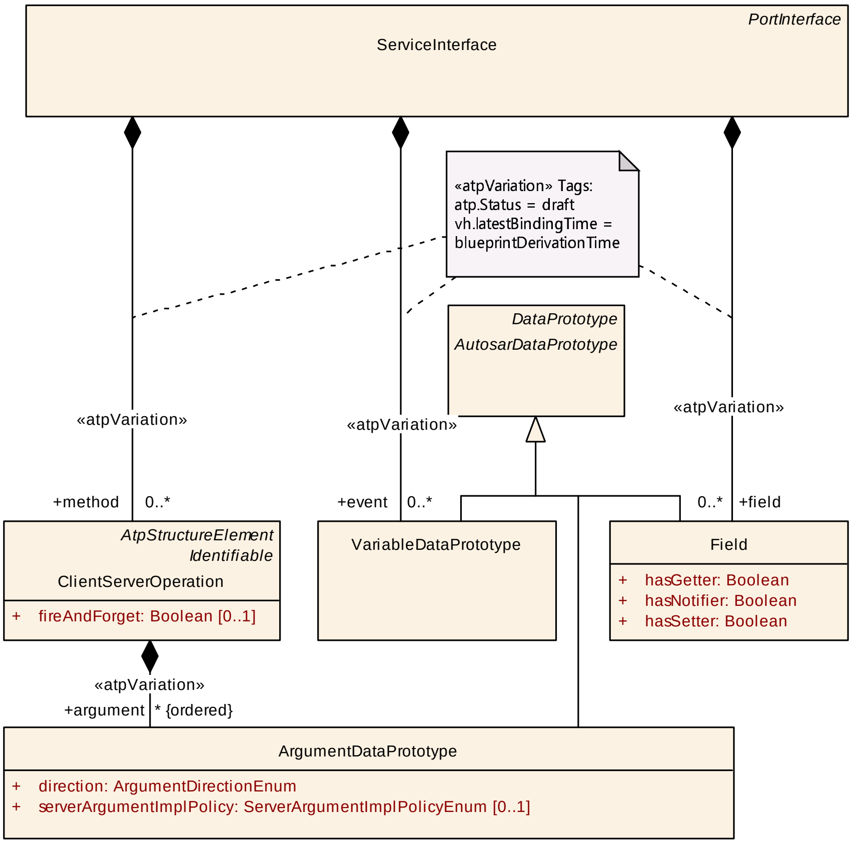


Figure 3.16: Modeling of the `ServiceInterface`

[constr_1483]{DRAFT} **Applicability of a `ServiceInterface`** ⌈ The applicability of a `ServiceInterface` shall be limited to the *AUTOSAR adaptive platform*, i.e. a `ServiceInterface` shall only be taken to type a `PortPrototype` if the latter is aggregated by an `AdaptiveApplicationSwComponentType` or by a `CompositionSwComponentType` defined in the context of an `Executable`. ⌋*()*

Please note that on the *AUTOSAR adaptive platform* there are use-cases for the utilization of a `ServiceInterface` **without** the existence of a corresponding `PortPrototype`. For more explanation, please refer to [TPS_MANI_01032].

Class	**ServiceInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This represents the ability to define a PortInterface that consists of a heterogeneous collection of methods, events and fields. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
event	VariableDataPrototype	*	aggr	This represents the collection of events defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime
field	Field	*	aggr	This represents the collection of fields defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime
method	ClientServerOperation	*	aggr	This represents the collection of methods defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime

Table 3.21: ServiceInterface

[TPS_MANI_01007]{DRAFT} **Atomic unit of service discovery** ⌈ As far as the application level is concerned, the atomic unit for **service discovery** on the *AUTOSAR adaptive platform* is the `ServiceInterface`. ⌋*(RS_MANI_00003)*

Please note that there is no obligation to have any `method`, `event`, or `field` defined in the context of a given `ServiceInterface`. In other words, the existence of a `ServiceInterface` by itself represents a valid semantics that has a value on its own.

For example, a use case could exist where a given service instance that corresponds to such a `ServiceInterface` is offered with the mere intention to signal that the ECU that provides the service instance is becoming ready for something, e.g. being diagnosed.

A tester could then take the existence of the offer as an indication to initiate a connection to the respective ECU.

3.4.2 Event

[TPS_MANI_01033]{DRAFT} **Semantics of `ServiceInterface.event`** ⌈ An `event` represents an update to a piece of data. The server decides when to send this update and makes sure that the `event` has full control over the value.

The occurrence of an event is transmitted from a server to one or more client(s). ⌋ *(RS_MANI_00003)*

[constr_1494]{DRAFT} **Initial value for event** ⌈ An ServiceInterface.event shall **not** have an initValue. ⌋*()*

For the client, the only way to get access to the value of an event is to receive an update of the event from the server.

As mentioned in [constr_1494], the Server always has full control over the value of the event and when it is sent to clients. Therefore, the definition of an initValue is not necessary.

Class	**VariableDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableDataPrototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided. In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.			
Base	*ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, Multilanguage Referrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table 3.22: VariableDataPrototype

3.4.3 Field

[TPS_MANI_01034]{DRAFT} **Semantics of ServiceInterface.field** ⌈ A field represents a piece of data hosted by a server that exposes to one or more client(s) a get accessor and/or a set mutator.

Clients can optionally receive notifications of changes of the field's value. ⌋ *(RS_MANI_00003)*

In comparison to an event, a field has a concrete value at any time. This conceptual difference can be explained along the following examples:

Let a traffic-sign detection be an example for the semantics of an event. The detection of a traffic-sign represents a discrete event in time that would be raised by the service component any time a speed limit sign is detected.

On the other hand, let a temperature preset of the in-vehicle air-condition be an example for a field that has a concrete value at any given time. The concrete value can be set by a client, can be obtained on request of a client, and – at the same time – a change of the temperature preset represents a relevant information by itself.

In summary, this means that if a field is defined with hasNotifier and a client subscribes to it then the current value of the field is sent back immediately to the subscriber in an event-like notification pattern as soon as the subscription to the field becomes effective.

Additional update notifications will be sent to subscribers whenever the value of the field gets updated.

In more technical terms, the get() accessor method the current field value can be retrieved by the client. By means of calling the set() mutator method the field value can be updated by the client.

Please note that all features that a field provides are optional, given a fulfillment of [constr_1673]. In the ServiceInterface.field description it is defined whether the field supports the on-change-notification (hasNotifier), the get() accessor (hasGetter) or the set() mutator (hasSetter).

Admittedly, the concept of the field is roughly equivalent to an aggregation of an event with correlated get()/set() methods.

As far as the meta-model is concerned, the fact that a field shall have a concrete value at any time demands the **definition of an initial value** for the field. This aspect is clarified by [TPS_MANI_03212].

The existence of meta-class field as a first class citizen in the ServiceInterface expresses in addition to the existence of an individual event and individual methods that the two defined accessor/mutator methods get() and set() are applied to the **same data object** and that the defined field notifier reports each value change of this data object to subscribers.

In other words, the semantics of meta-class Field is fully determined by the attributes hasGetter, hasSetter, and hasNotifier.

Therefore, a Field where all of these attributes are set to False wouldn't have any useful meaning and shall therefore not exist.

[constr_1673]{DRAFT} **Existence of attributes hasGetter, hasSetter, and hasNotifier** ⌈ For any given Field, **at least** one of the attributes

- hasGetter
- hasSetter
- hasNotifier

shall exist. ⌋*()*

Class	**Field**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to define a piece of data that can be accessed with read and/or write semantics. It is also possible to generate a notification if the value of the data changes. **Tags:** atp.Status=draft			
Base	*ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, Multilanguage Referrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hasGetter	Boolean	1	attr	This attribute controls whether read access is foreseen to this field.
hasNotifier	Boolean	1	attr	This attribute controls whether a notification semantics is foreseen to this field.
hasSetter	Boolean	1	attr	This attribute controls whether write access is foreseen to this field.

Table 3.23: Field

3.4.4 Method

[TPS_MANI_01035]{DRAFT} **Semantics of `ServiceInterface.method`** ⌈ A `method` represents a function that is executed by and in the scope of a server on request of one or more client(s). ⌋*(RS_MANI_00003)*

Class	**ClientServerOperation**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
fireAndForget	Boolean	0..1	attr	This attribute defines whether this method is a fire&forget method (true) or not (false). **Tags:** atp.Status=draft
possibleApError	ApApplicationError	*	ref	This reference identifies AdaptivePlatformApplication Errors as a possible error raised by the enclosing Client ServerOperation. **Tags:** atp.Status=draft
possibleApError Set	ApApplicationErrorSet	*	ref	This reference represents the ability to refer to an entire group of ApApplicationErrors as one model element instead of having to refer to all the represented Ap ApplicationErrors separately. **Tags:** atp.Status=draft

Table 3.24: ClientServerOperation

3.4.4.1 Fire and Forget Method

A so-called "fire & forget" method represents a special form of a method dedicated to the sole purpose of conveying information from a client to a server.

There is no expectation that the implementation of the method executes any kind of algorithm other than to merely accept the incoming data.

Spun from this angle, the semantics of a "fire & forget" method is comparable to the semantics of an event, only reverse.

In other words, the "fire & forget" method conveys the data and the occurrence of the data **from a client to a server**. For comparison, the event is used to convey information in combination with the occurrence of the information from **a server to a client**.

The *occurrence* aspect of this statement has the consequence that e.g. the number of "fire & forget" calls can be counted by the implementation of the server and this meta-information could be taken to convey additional semantics on top of the actual data.

[TPS_MANI_01064]{DRAFT} **Semantics of attribute method.fireAndForget** ⌈ The activation of the "fire & forget" semantics of a given method is achieved by setting the value of attribute method.fireAndForget to value true. ⌋*(RS_MANI_00003)*

[TPS_MANI_03118]{DRAFT} **Semantics of ServiceInterface.method with fireAndForget set to true** ⌈ A method with fireAndForget set to the value true represents a void-return-method where the client is not expecting any kind of acknowledge or handshake from the server side. ⌋*(RS_MANI_00003)*

[constr_3374]{DRAFT} **method with attribute fireAndForget set to true shall not have any inout or out arguments** ⌈ A method that has attribute fireAndForget set to the value true is not allowed to have any arguments with direction inout or out. ⌋*()*

[constr_3375]{DRAFT} **method with attribute fireAndForget set to true shall not reference an ApApplicationError** ⌈ A method that has attribute fireAndForget set to the value true is not allowed to reference an ApApplicationError in role possibleApError. ⌋*()*

[TPS_MANI_03119]{DRAFT} **Default value for the attribute fireAndForget of meta-class ClientServerOperation** ⌈ If the attribute fireAndForget is not defined then it shall be assumed that no "fire & forget" semantics is intended. ⌋*(RS_MANI_00003)*

3.4.5 Compatibility of Service Interfaces

This chapter defines ServiceInterface compatibility rules on the Application Design level that is independent of the later used transport layer.

Each transport layer mechanism (e.g. SOME/IP) may define its own compatibility rules. Therefore for each individual transport layer an own impact assessment on the compatibility needs to be performed whether the changed service interface has an incompatible representation on this transport layer.

The compatibility depends on the features that are used on the transport layer. For example, in SOME/IP a length field that is put in front of a struct allows that during deserialization unknown elements at the end of an extensible data struct are skipped.

An additional option in SOME/IP is the usage of Data IDs in front of optional struct members. With this approach the receiver can skip unknown members of the struct, i.e. where the Data ID is unknown.

Therefore on the Application Design level all changes of data types shall be handled carefully since only the used transport layer and the used features on the transport layer decide whether the change is compatible or not.

[constr_3387]{DRAFT} **Compatibility of `PortPrototype`s of different `ServiceInterface`s** ⌈ `PortPrototype`s of different `ServiceInterface`s are compatible if:

- For each `event` defined in the context of the `ServiceInterface` of the required `PortPrototype` a compatible `event` exists in the `ServiceInterface` of the provided `PortPrototype` according to [constr_3388].
- For each `method` defined in the context of the `ServiceInterface` of the required `PortPrototype` a compatible `method` exists in the `ServiceInterface` of the provided `PortPrototype` according to [constr_3389].
- For each `field` defined in the context of the `ServiceInterface` of the required `PortPrototype` a compatible `field` exists in the `ServiceInterface` of the provided `PortPrototype` according to [constr_3390].

⌋*()*

[constr_3388]{DRAFT} **Compatibility of `events`** ⌈ Two `event`s are assumed as compatible if the following conditions apply:

- the two `event`s have identical `shortName`s.

⌋*()*

[constr_3389]{DRAFT} **Compatibility of `methods`** ⌈ Two `method`s are assumed as compatible if the following conditions apply:

- the two `method`s have identical `shortName`s.
- the two `method`s have the same number of `ArgumentDataPrototype`s.

⌋*()*

[constr_3390]{DRAFT} **Compatibility of `fields`** ⌈ Two `field`s are assumed as compatible if the following conditions apply:

- the two fields have identical shortNames.
- if the attribute hasNotifier is set to true for the field defined in the context of the ServiceInterface of the required PortPrototype then the hasNotifier attribute in the field that is defined in the context of the ServiceInterface of the provided PortPrototype shall be true as well.
- if the attribute hasGetter is set to true for the field defined in the context of the ServiceInterface of the required PortPrototype then the hasGetter attribute in the field that is defined in the context of the ServiceInterface of the provided PortPrototype shall be true as well.
- if the attribute hasSetter is set to true for the field defined in the context of the ServiceInterface of the required PortPrototype then the hasSetter attribute in the field that is defined in the context of the ServiceInterface of the provided PortPrototype shall be true as well.

⌋*()*

Please note that the constraints [constr_3388], [constr_3389] and [constr_3390] do not make any statements about the compatibility of AutosarDataTypes of the AutosarDataPrototypes.

Finally the compatibility rules of the used transport layer will decide whether two ServiceInterfaces are compatible or not. The constraints defined in this chapter define a basic set of rules that are valid for all supported transport layers.

If one wants to make sure that two AutosarDataPrototypes inside of a ServiceInterface are compatible then both AutosarDataPrototypes shall be typed by an identical AutosarDataType.

3.4.5.1 Modeling of ServiceInterface versions

With constraint [constr_3387] a ServiceInterface can be updated based on the requirements of one or more consumers, which can start using this new ServiceInterface immediately.

The other consumers of this service do not need to switch to using the latest version of this ServiceInterface, but can continue to use older versions of the ServiceInterface they were designed for and tested with.

Using multiple versions of the same ServiceInterface supports an independent life cycle of services and allows to change and enhance ServiceInterfaces without affection of existing consumers. This chapter describes how different versions of the same ServiceInterface can be modeled.

A version of a ServiceInterface may be defined for example as ServiceInterface with an own shortName (e.g. Service_Version1, Service_Version2) or as

ServiceInterface that is located in an own ARPackage (e.g. /Version1/Service, /Version2/Service).

It is also allowed to assign a different *namespace* to the different ServiceInterface versions to influence the generated code, e.g. to generate com::version1::Service and com::version2::Service.

During the ServiceInterfaceDeployment the ServiceInterface is mapped to a middleware transport layer where the necessary middleware transport layer specific configuration settings are performed, as described in chapter 9.1.

For example it is possible to assign the same SOME/IP serviceInterfaceId to different versions of the same ServiceInterface, but a different majorVersion or minorVersion.

This approach takes into account that the compatibility of ServiceInterfaces is heavily influenced by the used transport binding.

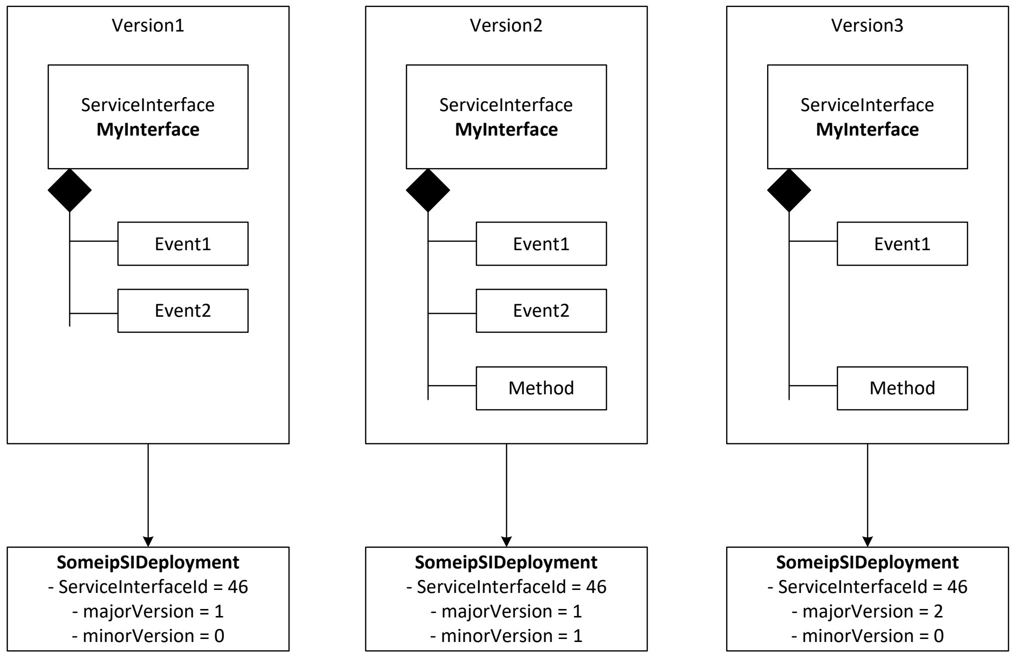


Figure 3.17: Example for different versions of the same ServiceInterface

Please note that the compatibility rules for SOME/IP are described in [8].

3.4.6 Namespace

The definition of a ServiceInterface has a direct impact on the code of an application on the *AUTOSAR adaptive platform*.

Without going into too much detail at this point, it is necessary to support the definition of a *namespace* in the context of a ServiceInterface.

The namespace shall be used to encapsulate source code related to the ServiceInterface and thus avoid name clashes with the content of other definitions of ServiceInterfaces.

In principle, the definition of the namespace around a concrete ServiceInterface could be derived from the structure of ARPackages in which the definition of the ServiceInterface is contained. However, this approach puts some constraints of the package structure.

The same ServiceInterface may be used in different projects that may or may not demand the usage of a specific *different* package structure.

This placement of the same ServiceInterface in potentially different package hierarchies would lead to the definition of different namespaces, and thus the necessity to create or generate the code representing the ServiceInterface **plus** the code that uses this definition again and again.

One way to overcome this potential issue is to attach a dedicated namespace definition to the definition of the ServiceInterface itself.

This approach is documented in Figure 3.18.

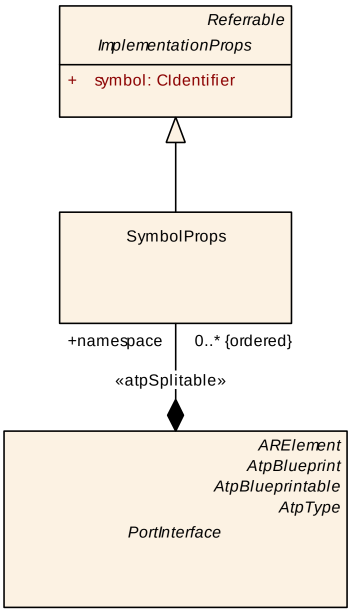


Figure 3.18: Specification of namespaces in PortInterfaces

[TPS_MANI_01004]{DRAFT} **Semantics of ServiceInterface.namespace** ⌈ The aggregation ServiceInterface.namespace shall be used to define the namespace to be used for the source code that corresponds to the given ServiceInterface. ⌋ *(RS_MANI_00003)*

[TPS_MANI_01005]{DRAFT} **The definition of the namespace of a ServiceInterface may follow a hierarchical pattern** ⌈ The namespace of a ServiceInterface may follow a hierarchical pattern, as supported by many modern programming languages.

The separator between the elements of the hierarchical namespace definition depends on the used programming language and is not explicitly defined in the model.

The model only defines the elements of the hierarchical namespace pattern. ⌋ *(RS_MANI_00003)*

As the consequence of the ability to define a hierarchical namespace, the aggregation `ServiceInterface.namespace` is qualified as being `ordered`.

This means that the order of individual elements to the collection of `namespace`s has a semantical relevance[5].

[TPS_MANI_01006]{DRAFT} **Ordered definition of `ServiceInterface.namespace`** ⌈ In a hierarchical definition of `ServiceInterface.namespace` the order of `namespace` fragments shall be maintained in the translation of the namespace to source code.

In other words, the first `namespace` fragment shall appear first, followed by the second `namespace` fragment, and so on. ⌋*(RS_MANI_00003)*

Listing 3.3: Example for the definition of a namespace for a given `ServiceInterface`

```
<SERVICE-INTERFACE>
  <SHORT-NAME>MyServiceInterface</SHORT-NAME>
  <NAMESPACES>
    <SYMBOL-PROPS>
      <SHORT-NAME>first</SHORT-NAME>
      <SYMBOL>com</SYMBOL>
    </SYMBOL-PROPS>
    <SYMBOL-PROPS>
      <SHORT-NAME>second</SHORT-NAME>
      <SYMBOL>myCompany</SYMBOL>
    </SYMBOL-PROPS>
    <SYMBOL-PROPS>
      <SHORT-NAME>third</SHORT-NAME>
      <SYMBOL>software</SYMBOL>
    </SYMBOL-PROPS>
  </NAMESPACES>
</SERVICE-INTERFACE>
```

Class	***PortInterface*** (abstract)
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
Note	Abstract base class for an interface that is either provided or required by a port of a software component.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Subclasses	ClientServerInterface, *DataInterface*, *DiagnosticPortInterface*, ModeSwitchInterface, *PersistencyInterface*, *PlatformHealthManagementInterface*, RestServiceInterface, ServiceInterface, *TimeSynchronizationInterface*, TriggerInterface

▽

[5]This means that the definition of a namespace `a::b` is semantically different from the definition of a namespace `b::a`.

△

Class	***PortInterface*** (abstract)			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
namespace (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft

Table 3.25: PortInterface

Class	**SymbolProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This meta-class represents the ability to contribute a part of a namespace.			
Base	*ARObject*, *ImplementationProps*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.26: SymbolProps

The Listing 3.3 exemplifies the statement made by [TPS_MANI_01006], i.e. the resulting name space in e.g. C++ would look like sketched in Listing 3.4.

```
namespace com {
    namespace myCompany {
        namespace software {

        }
    }
}
```

Listing 3.4: Resulting namespace for the example `ServiceInterface`

3.4.7 Error Handling

The modeling of error handling on the *AUTOSAR adaptive platform* slightly differs from the approach implemented on the *AUTOSAR classic platform*.

In particular, the formal representation of an error during the execution of a `method` is done in a global scope, i.e. such a definition can be reused arbitrarily by any `ServiceInterface`.

[TPS_MANI_01190]{DRAFT} **Semantics of `ApApplicationError`** ⌈ Meta-class `ApApplicationError` represents the ability to define the existence of an error during the execution of a `method` independently of the scope of a `ServiceInterface` or `ClientServerOperation`. ⌋*()*

Class	ApApplicationError			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to formally specify the semantics of an application error on the AUTOSAR adaptive platform **Tags:** atp.Status=draft atp.recommendedPackage=ApplicationErrors			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
errorCode	Integer	1	attr	This attribute has the ability to specify the error code value within the enclosing AdaptivePlatformApplicationError.
errorDomain	ApApplicationErrorDomain	1	ref	This reference represents the error domain of the ApApplicationError. **Tags:** atp.Status=draft

Table 3.27: ApApplicationError

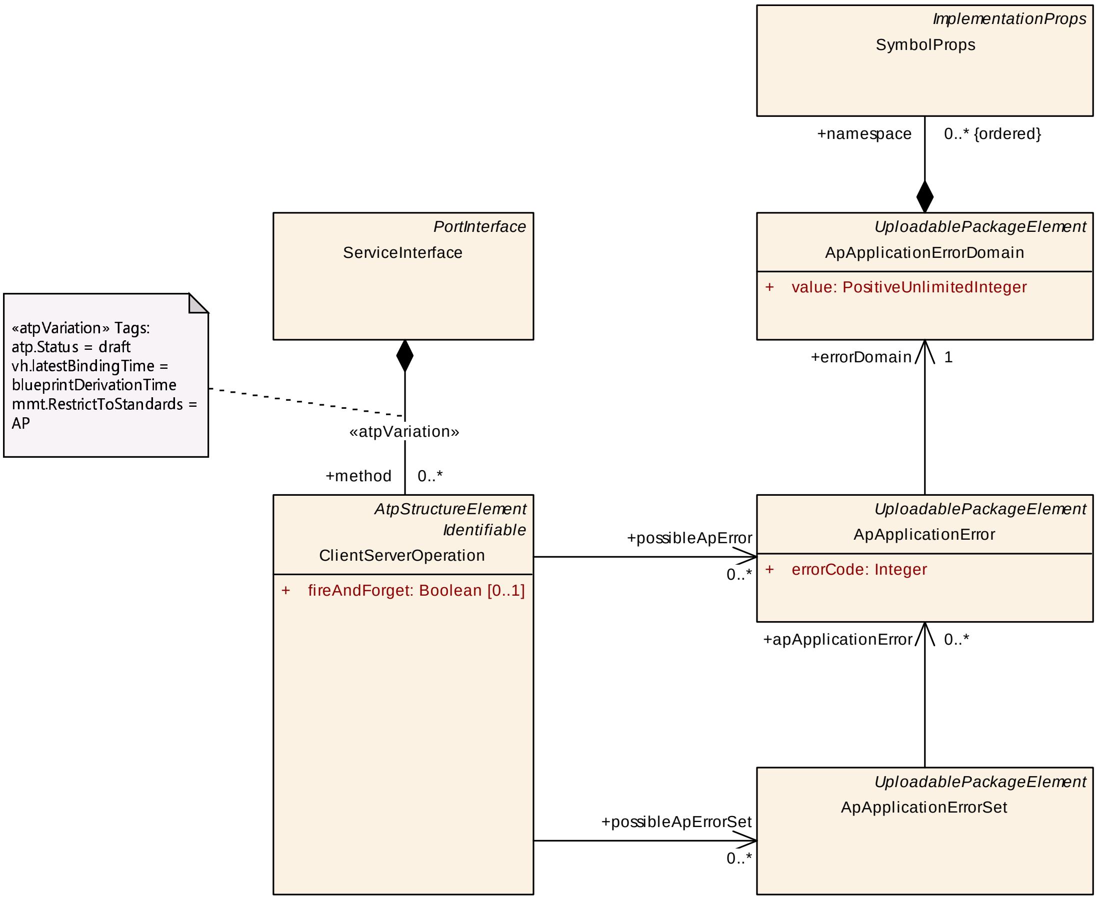


Figure 3.19: Modeling of `ApApplicationError` on the *AUTOSAR adaptive platform*

[TPS_MANI_01198]{DRAFT} **Semantics of `ApApplicationErrorSet`** ⌈ Meta-class `ApApplicationErrorSet` has the ability to group references to `ApApplicationError` and thus represents a “proxy” to this group of references towards the `ClientServerOperation`.

The use case for this modeling ability is that some ClientServerOperations may have to reference an identical significant number of ApApplicationErrors.

Letting each of the ClientServerOperations repeat the same set of references to ApApplicationError is considered unnecessary and therefore the ability to refer to a group instead of individual references is provided as an alternative. ⌋ *(RS_MANI_00026)*

The decision whether or not an ApApplicationErrorSet is defined an referenced from specific ClientServerOperations has to be done on an individual basis. AUTOSAR just wants to make this business as straightforward as possible.

Please note that it is also positively possible to mix the usage of ClientServerOperation.possibleApError and ClientServerOperation.possibleApErrorSet.

Class	**ApApplicationErrorSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class acts as a reference target that represents an entire collection of APApplicationErrors. This takes the burden from ClientServerOperations that reference a larger number of ApApplication Errors. **Tags:** atp.Status=draft atp.recommendedPackage=ApplicationErrorSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
apApplication Error	ApApplicationError	*	ref	Thi reference represents the collection of ApApplication Error represented by the enclosing ApApplicationErrorSet **Tags:** atp.Status=draft

Table 3.28: ApApplicationErrorSet

As ApApplicationError is no longer defined within the scope of a ServiceInterface, there is no need to define a mapping between two ApApplicationErrors by means of a dedicated sub-class of ServiceInterfaceElementMapping.

[TPS_MANI_01191]{DRAFT} **Modeling of possible errors** ⌈ A ClientServerOperation aggregated by a ServiceInterface in the role method shall reference

- one or more ApApplicationError(s) in the role possibleApError
- one or more ApApplicationErrorSet(s) in the role possibleApErrorSet

to formally specify the existence of possible errors raised by the ClientServerOperation. ⌋*(RS_MANI_00026)*

[TPS_MANI_01192]{DRAFT} **Semantics of ApApplicationErrorDomain** ⌈ Meta-class ApApplicationErrorDomain shall be used to define a specific error domain that can potentially be standardized by AUTOSAR.

Therefore, the definition of such an error domain is not defined in the scope of the ApApplicationError itself. Instead, an ApApplicationError identifies the applicable error domain by means of a reference in the role errorDomain.

It is possible to attach the definition of a namespace to ApApplicationErrorDomain because this information is relevant for the language binding. ⌋ *(RS_MANI_00026)*

Class	**ApApplicationErrorDomain**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to define a global error domain for an ApApplicationError. **Tags:** atp.Status=draft atp.recommendedPackage=ApplicationErrorDomains			
Base	*ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable, UploadablePackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
namespace (ordered)	SymbolProps	*	aggr	This aggregation defines the namespace of the Ap ApplicationErrorDomain **Tags:** atp.Status=draft
value	PositiveUnlimitedInteger	1	attr	This attribute identifies the error category.

Table 3.29: ApApplicationErrorDomain

[constr_1627]{DRAFT} **Supported value range for attribute ApApplicationErrorDomain.value** ⌈ The supported value range of attribute ApApplicationErrorDomain.value is limited to the interval [0..18446744073709551616]. ⌋ *()*

[constr_1625]{DRAFT} **Existence of reference ApApplicationError.errorDomain** ⌈ For each ApApplicationError, the reference errorDomain shall exist.

In other words, the association of an ApApplicationError with a corresponding ApApplicationErrorDomain is mandatory. ⌋ *()*

[constr_1664]{DRAFT} **Unique ApApplicationError.shortName** ⌈ Within the set of all ApApplicationErrors that reference a given ApApplicationErrorDomain in the role errorDomain the attribute ApApplicationError.shortName shall have a unique value. ⌋ *()*

[constr_1665]{DRAFT} **Unique ApApplicationError.errorCode** ⌈ Within the set of all ApApplicationErrors that reference a given ApApplicationErrorDomain in the role errorDomain the attribute ApApplicationError.errorCode shall have a unique value. ⌋ *()*

Rationale for the existence of [constr_1664] and [constr_1665]: the language binding for C++ foresees the usage of attributes ApApplicationError.shortName and ApApplicationError.errorCode for the creation of an enum within the context of the ApApplicationErrorDomain.

Duplicates in terms of labels of enumerators or values of enumerators lead to compile-time errors.

3.4.8 Service Interface Data Type Mapping

An important step in the workflow of implementing software on the *AUTOSAR adaptive platform* is the creation of a code-based representation of a ServiceInterface to make it accessible for the application code.

This creation of a code-based representation is usually automatized and will be executed by a code generator. This code generator needs an input from the model. The main input for this purpose is obviously the definition of the ServiceInterface itself.

However, this is not sufficient. The designer of a ServiceInterface is free to use ApplicationDataTypes for the specification of the details of the ServiceInterface.

It is therefore necessary to provide the definition of an AbstractImplementationDataType for each of the used ApplicationDataType. In the meta-model, this correspondence is implemented by means of the meta-class DataTypeMappingSet[6].

However, from the methodological point of view it is considered inappropriate to let ServiceInterface directly refer to one or more DataTypeMappingSet(s).

For clarification, this would mean that the mapping of ApplicationDataType to AbstractImplementationDataType becomes an integral part of the definition of the ServiceInterface although the mapping itself does not really contribute to the actual semantics of the ServiceInterface.

As a consequence, the ServiceInterface would have to be updated whenever the mapping between data types changes.

But since the definition of ServiceInterfaces are usually considered very stable a frequent update for the mere purpose of acknowledging a change in the data type mapping is not acceptable.

In this concrete case, the described problem can be circumvented by the definition of a mapping class that refers to both a ServiceInterface and a DataTypeMappingSet and therefore create the correspondence without the need to update the ServiceInterface.

Although the prelude into this chapter suggests the existence of a meta-class that maps a ServiceInterface to one or more DataTypeMappingSet(s) the actual meta-model is designed with a broader focus.

In the future, there could be further kinds of PortInterfaces beside the ServiceInterface that need to fulfill the same use case.

Consequently, the name of the meta-class created for this purpose is PortInterfaceToDataTypeMapping.

[6] For more background regarding the definition and use of meta-class DataTypeMappingSet please refer to [1].

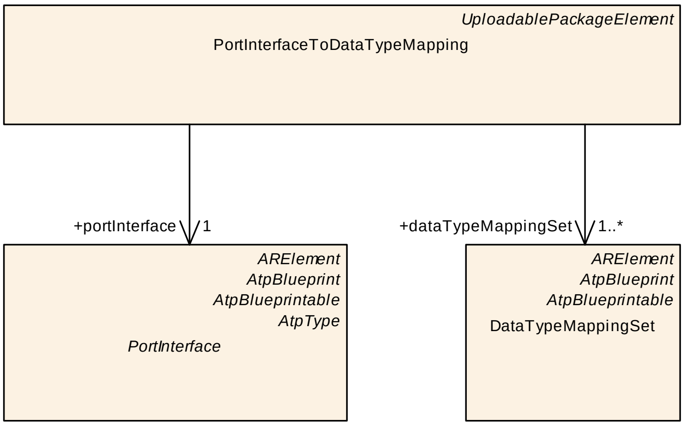


Figure 3.20: Modeling of `PortInterfaceToDataTypeMapping`

[constr_1507]{DRAFT} **`PortInterfaceToDataTypeMapping` is only applicable to `ServiceInterface` or `PersistencyKeyValueDatabaseInterface`** ⌈ `PortInterfaceToDataTypeMapping.portInterface` shall only refer to **either** a `ServiceInterface` **or** a `PersistencyKeyValueDatabaseInterface`. ⌋*()*

Class	**PortInterfaceToDataTypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to associate a PortInterface with a DataTypeMappingSet. This association is needed for the generation of header files in the scope of a single PortInterface. The association is intentionally made outside the scope of the PortInterface itself because the designers of a PortInterface most likely will not want to add details about the level of ImplementationDataType. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaceToDataTypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataTypeMappingSet	DataTypeMappingSet	1..*	ref	This represents the reference to the applicable data TypemappingSet **Tags:** atp.Status=draft atp.StatusComment=Reserved for adaptive platform
portInterface	PortInterface	1	ref	This represents the reference to the applicable Port Interface **Tags:** atp.Status=draft atp.StatusComment=Reserved for adaptive platform

Table 3.30: PortInterfaceToDataTypeMapping

Class	**DataTypeMappingSet**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes
Note	This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes. In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups. **Tags:** atp.recommendedPackage=DataTypeMappingSets
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	**DataTypeMappingSet**			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an Application DataType and its AbstractImplementationDataType.
modeRequest TypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an Mode DeclarationGroup and its AbstractImplementationData Type.

Table 3.31: DataTypeMappingSet

Class	**DataTypeMap**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents the relationship between ApplicationDataType and its implementing Abstract ImplementationDataType.			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
applicationData Type	ApplicationDataType	1	ref	This is the corresponding ApplicationDataType
implementation DataType	AbstractImplementation DataType	1	ref	This is the corresponding AbstractImplementationData Type.

Table 3.32: DataTypeMap

3.5 Service Interface Mapping

Please note that, according to [TPS_MANI_01007], the `ServiceInterface` becomes the single basis for both `VFB`-based and *external* (i.e. using communication networks) communication.

This concept is in stark contrast to the approach on the *AUTOSAR classic platform* where different model elements are used for the `VFB`-level (`PortInterface`) and the network-level (`SystemSignal`, `ISignal`, and `ISignalIPdu`).

The usage of different model elements optimally supports the existence of different granularity for `VFB`-based vs. network-based communication.

In other words, design of communication on the network level may be subject to different design restrictions, e.g. keep the bus load caused by service discovery manageable by defining coarse-grained communication packages.

Opposed to that, designers on the `VFB` level may want to define interface granularity to achieve maximum reusability.

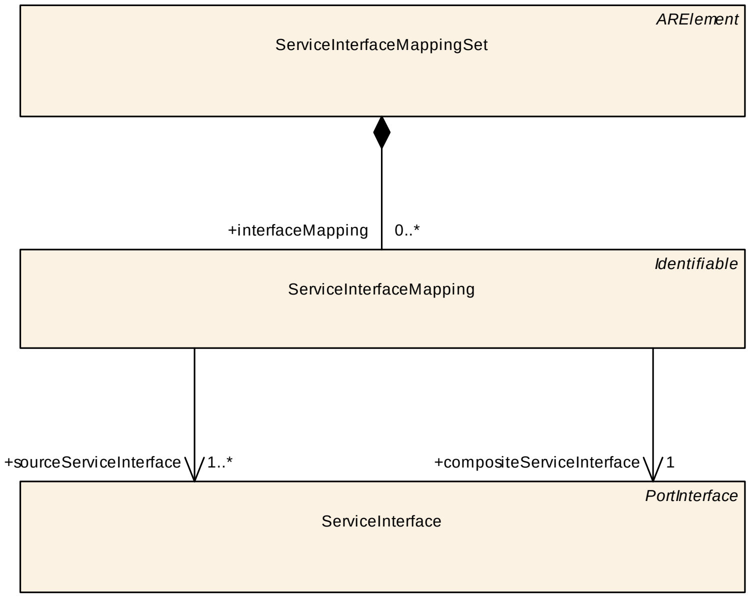


Figure 3.21: Modeling of the ServiceInterfaceMapping

[TPS_MANI_01002]{DRAFT} **Semantics of meta-class ServiceInterfaceMapping** ⌈ In order to sort out a potentially different motivation between the definition of

- ServiceInterfaces explicitly designed for VFB-based communication and
- ServiceInterfaces explicitly designed for network-based communication

meta-class ServiceInterfaceMapping is available to map

- (fine-grained) ServiceInterfaces for the VFB-communication to
- (coarse-grained) ServiceInterfaces for network communication.

⌋*(RS_MANI_00017)*

[TPS_MANI_01032]{DRAFT} **Usage of ServiceInterfaceMapping** ⌈ The ability to apply a ServiceInterfaceMapping can be used in two different ways:

- It is possible to derive a dedicated AdaptiveApplicationSwComponentType that implements the mapping functionality. A SwComponentPrototype derived from this so-called *facade* software-component would expose PortPrototypes for each of the ServiceInterfaces.

  Other SwComponentPrototypes could then "connect" to the PortPrototypes typed by ServiceInterfaces referenced in the role sourceServiceInterface.

  The PortPrototype typed by the ServiceInterface referenced in the role compositeServiceInterface is used for external communication.

- It is also possible to configure the communication middleware to offer or require a service typed by the ServiceInterface referenced in the role compositeServiceInterface.

A configuration of the relevant ids for this scenario is possible as part of the Execution Manifest.

⌋(*RS_MANI_00017*)

Figure 3.22 summarizes the idea behind the creation of a *facade* software-component. The latter is able to “bundle” the communication of different PortPrototypes owned by potentially different SwComponentTypes for external communication.

In other words, elements event1 owned by SWC1 and event2 owned by SWC1 are combined into one ServiceInterface used to type one PortPrototype of the *facade* software-component.

From the communication-related outside point-of-view, SWC3 acts like a facade to the “inner structure” created by SWC1 and SWC2 that is, by way of the existence of SWC3, abstracted away.

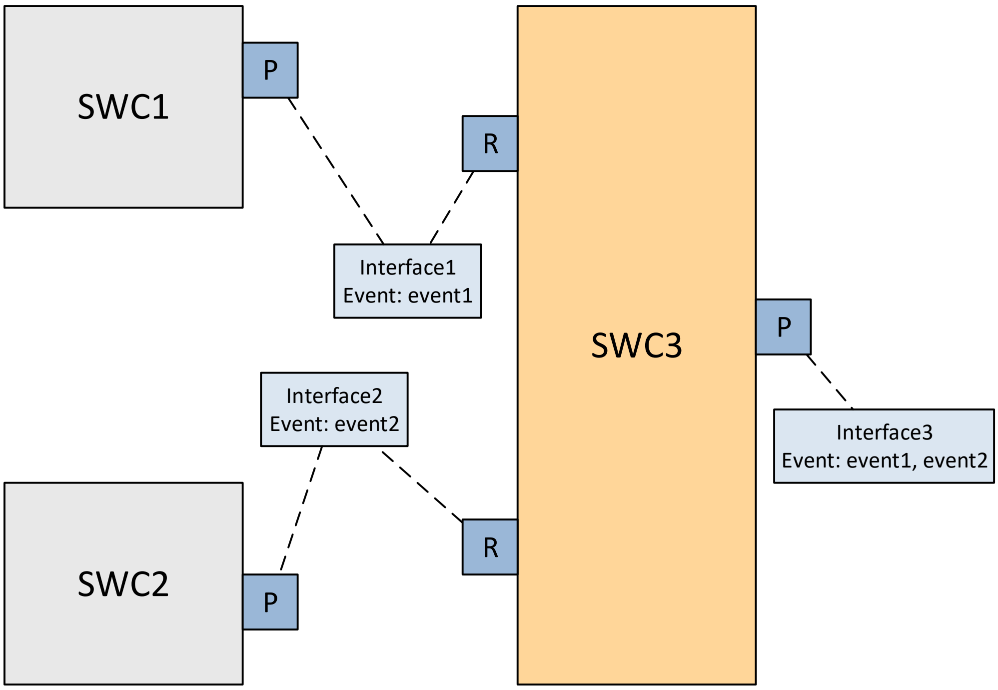


Figure 3.22: Concept of a facade software-component

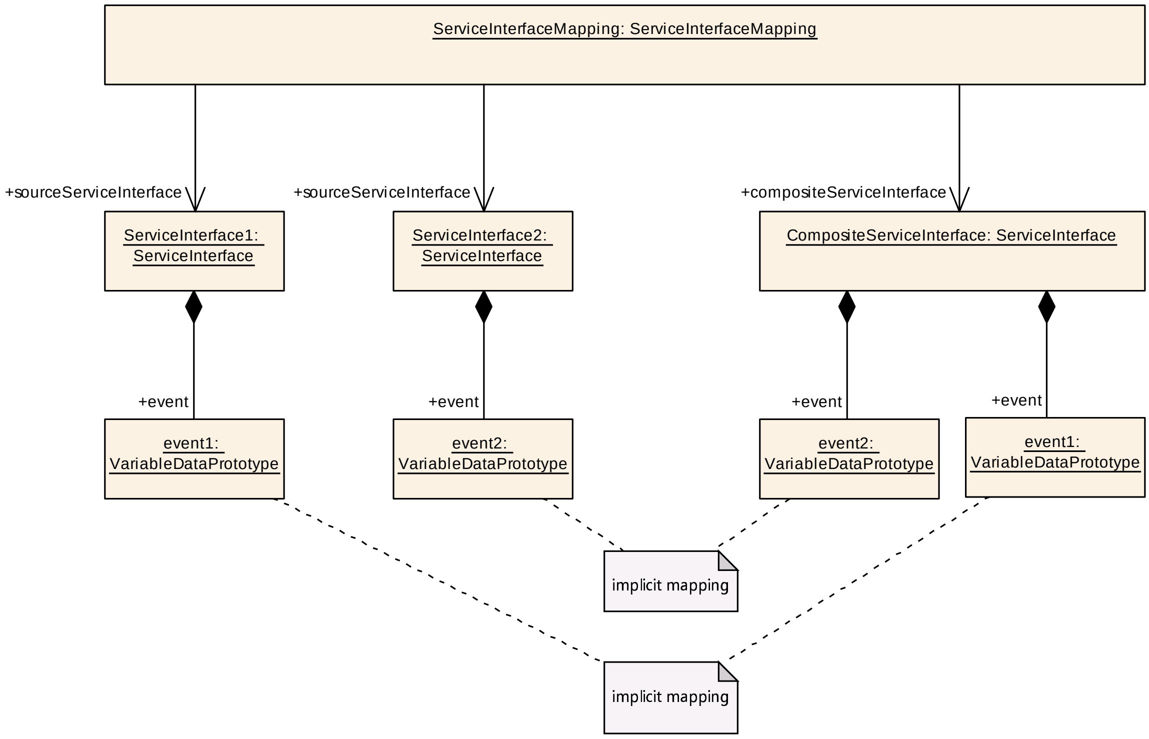


Figure 3.23: Example for the application of a ServiceInterfaceMapping

[TPS_MANI_01022]{DRAFT} **Concept behind ServiceInterfaceMapping** ⌈ The concept behind the definition of a ServiceInterfaceMapping is that **all elements** of the sourceServiceInterface are required to have a **counterpart of the same kind** (ServiceInterface.event, ServiceInterface.field, or ServiceInterface.method) and with the identical shortName. ⌋*(RS_MANI_00017)*

The regulation stated in [TPS_MANI_01022] is exemplified in Figure 3.23.

Please note that the creation of a ServiceInterfaceMapping is considered an atomic step, it is unlikely that such a ServiceInterfaceMapping is partially created and then later finished by a different party.

After all, there are mutually exclusive ways to specify the mapping, and any creator of a partial mapping of ServiceInterfaces could not be sure which of the alternatives apply for a specific pairing of one ServiceInterface with another without already knowing the other ServiceInterface (in which case the mapping can already be completed).

Therefore, there is no need to set the lower multiplicity of the references to ServiceInterface to 0.

Class	ServiceInterfaceMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	Specifies one ServiceInterfaceMapping that allows to define that a ServiceInterface is composite of several other ServiceInterfaces. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
compositeServiceInterface	ServiceInterface	1	ref	This represents the composite ServiceInterface. **Tags:** atp.Status=draft
sourceServiceInterface	ServiceInterface	1..*	ref	ServiceInterface that is mapped into the composite ServiceInterface. **Tags:** atp.Status=draft

Table 3.33: ServiceInterfaceMapping

Class	ServiceInterfaceMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class represents the ability to aggregate a collection of ServiceInterfaceElementMappings. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaceMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
elementMapping	ServiceInterfaceElementMapping	*	aggr	This represents the collection of ServiceInterfaceElementMappings aggregated at the ServiceInterfaceElementMappingSet **Tags:** atp.Status=draft
interfaceMapping	ServiceInterfaceMapping	*	aggr	This represents the collection of ServiceInterfaceMappings owned by the ServiceInterfaceMappingSet. **Tags:** atp.Status=draft

Table 3.34: ServiceInterfaceMappingSet

[TPS_MANI_01003]{DRAFT} **Limitation of the applicability of ServiceInterfaceMapping** ⌈ The applicability of the ServiceInterfaceMapping is limited to cases where the shortNames of the elements of the compositeServiceInterface are **unique** in the context of the compositeServiceInterface. ⌋ *(RS_MANI_00017)*

As already indicated, the meta-class ServiceInterfaceMappingSet has been defined as a container for both ServiceInterfaceMappings as well as the ServiceInterfaceElementMapping introduced in section 3.6.

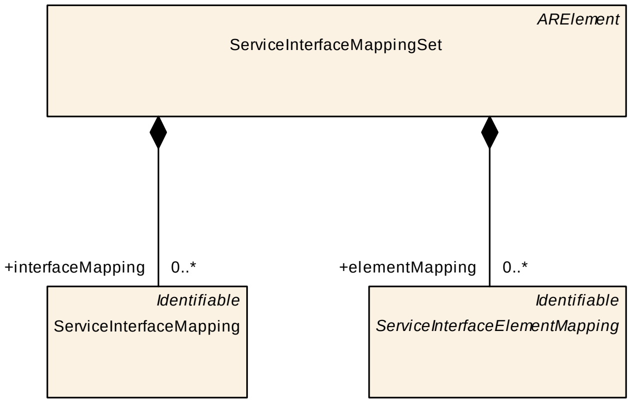


Figure 3.24: Modeling of the ServiceInterfaceMappingSet

Note that the ServiceInterfaceMapping is not an up-front association (by means of SwConnectors) between communication ends in the sense of section 3.4.5.

As stated in [TPS_MANI_01032], the ServiceInterfaceMapping allows for the derivation of a facade software-component or a proper configuration of the communication middleware.

The compatibility between the sourceServiceInterfaces and the compositeServiceInterface is achieved by an adequate transformation implemented in the facade software-component or the configuration of the middleware.

Thus, connecting ServiceInterfaces (or parts of them) via ServiceInterfaceMappings is not constrained by any compatibility rules apart from the ones stated in [TPS_MANI_01022].

3.6 Service Interface Element Mapping

3.6.1 Overview

The existence of the ServiceInterfaceMapping leaves the question about how ServiceInterfaces where elements have non-matching shortName can be mapped.

The answer to this question is provided by the ability to create an element-wise mapping of elements of the same kind.

Figure 3.25 provides an example of how such a mapping on element basis looks like. Note that, in this example, both ServiceInterface1 and ServiceInterface2 aggregate a field with the shortName field1.

This configurations disqualifies the scenario from the application of the ServiceInterfaceMapping, as of [TPS_MANI_01003]. The element-wise mapping, however, is able to work around the existence of the shortName field1 in both “source” ServiceInterfaces quite nicely:

- `ServiceInterface1.field1` is mapped to `CompositeServiceInterface.leftField`
- `ServiceInterface2.field1` is mapped to `CompositeServiceInterface.rightField`

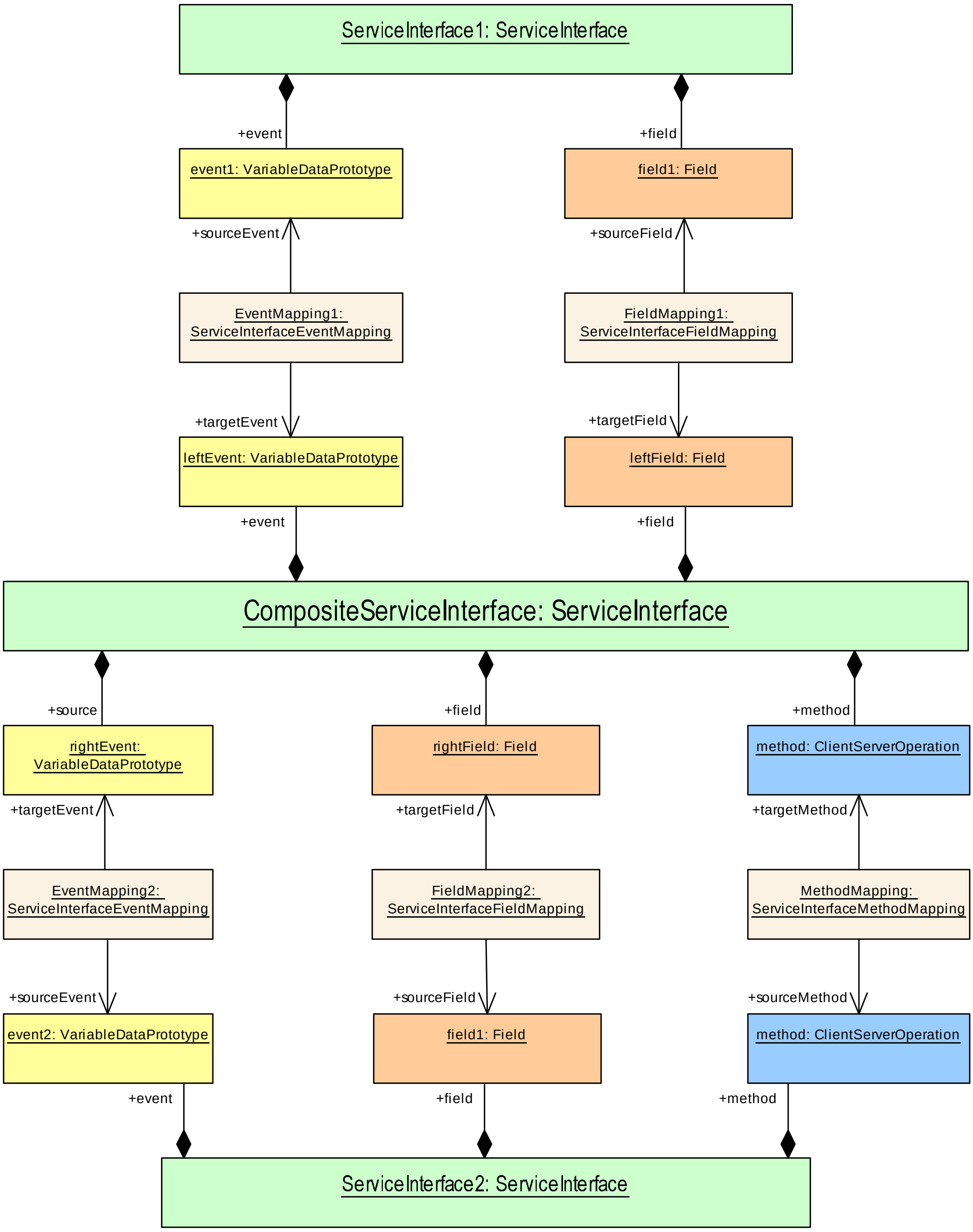


Figure 3.25: Example for a mapping of elements of `ServiceInterface`

The formal modeling of the individual mappings is described in section 3.6.

Please note that it is **not intended** to mix a mapping of ServiceInterfaces with a mapping of elements of a ServiceInterface.

In other words, as soon as a mapping between two ServiceInterfaces exists, it is not supported that a mapping between elements of the same pair of ServiceInterfaces exists. This important restriction is formalized by [constr_1482].

[constr_1482]{DRAFT} **Mapping of service interfaces vs. mapping of service interface elements** ⌈ In order to establish a mapping between a given pair of ServiceInterfaces, at most **one of** the following alternatives can exist:

- the given pair of ServiceInterfaces is referenced by a ServiceInterfaceMapping, where one ServiceInterface is referenced in the role sourceServiceInterface and the other ServiceInterface is referenced in the role compositeServiceInterface.
- an arbitrary mixture of the following options exists:
  - an event aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceEventMapping in the role sourceEvent and one events aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceEventMapping in the role targetEvent.
  - a field aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceFieldMapping in the role sourceField and one fields aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceFieldMapping in the role targetField.
  - a method aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceMethodMapping in the role sourceMethod and one methods aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceMethodMapping in the role targetMethod.

⌋*()*

Of course, it is possible that the same ServiceInterface is referenced by mappings to elements and mappings to entire ServiceInterfaces. The limitation formalized in [constr_1482] always applies to a **pair** of ServiceInterfaces.

A mapping between elements of ServiceInterfaces is modeled by means of a subclass of the abstract meta-class ServiceInterfaceElementMapping.

Class	*ServiceInterfaceElementMapping* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This abstract meta-class acts as base class for the mapping of specific elements of a ServiceInterface. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ServiceInterfaceEventMapping, ServiceInterfaceFieldMapping, ServiceInterfaceMethodMapping			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.35: ServiceInterfaceElementMapping

ServiceInterfaceElementMappings are aggregated by a ServiceInterfaceMappingSet that – in principle – allows for an arbitrary grouping of ServiceInterfaceElementMappings.

Please note that the creation of a ServiceInterfaceElementMapping is considered an atomic step, i.e. it is unlikely that such a ServiceInterfaceElementMapping is partially created, handed over to a different party and then later finished by that different party.

After all, there are mutually exclusive ways to specify the mapping, and any creator of a partial mapping of ServiceInterfaces could not be sure which of the alternatives apply for a specific pairing of one ServiceInterface with another without already knowing the other ServiceInterface (in which case the mapping can already be completed).

Therefore, there is no need to set the lower multiplicity of the references to elements of the ServiceInterface to 0.

3.6.2 Service Interface Event Mapping

[TPS_MANI_01024]{DRAFT} **Semantics of ServiceInterfaceEventMapping** ⌈
Meta-class ServiceInterfaceEventMapping has the ability to map a ServiceInterface.event referenced in the role sourceEvent explicitly to another ServiceInterface.event referenced in the role targetEvent. ⌋*(RS_MANI_00017)*

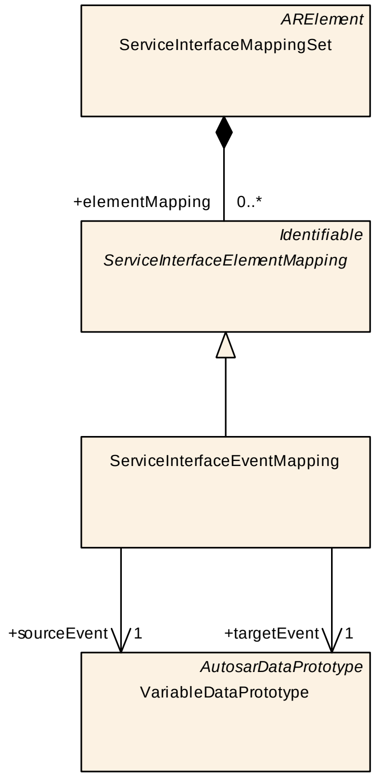


Figure 3.26: Modeling of the ServiceInterfaceEventMapping

Class	**ServiceInterfaceEventMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between events of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceElementMapping*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sourceEvent	VariableDataPrototype	1	ref	Reference to an event that is contained in the source ServiceInterface. **Tags:** atp.Status=draft
targetEvent	VariableDataPrototype	1	ref	Reference to an event that is contained in the composite ServiceInterface. **Tags:** atp.Status=draft

Table 3.36: ServiceInterfaceEventMapping

The explicit mapping implemented by ServiceInterfaceEventMapping does **not** require equal shortNames on both sides of the mapping.

It is also possible to map a given event of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetEvent, as exemplified by Figure 3.27.

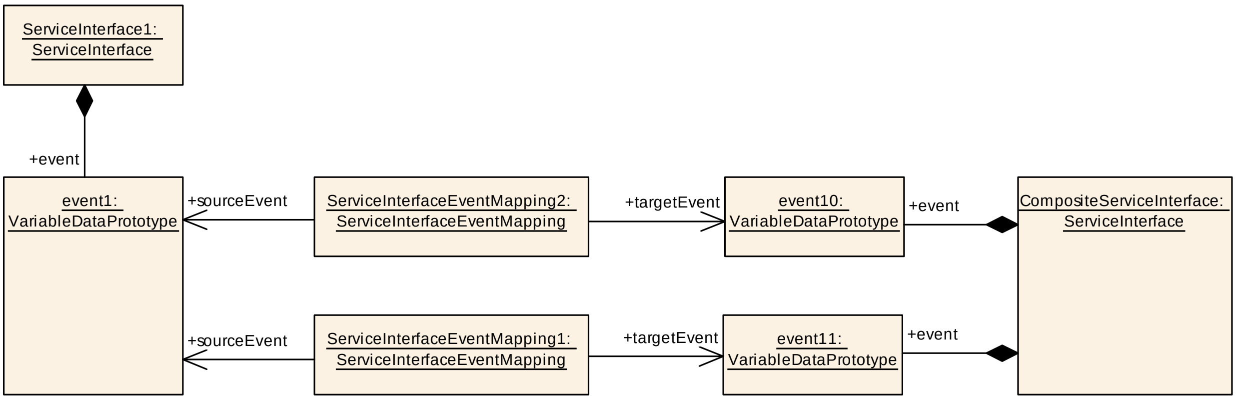


Figure 3.27: Example for the application of a ServiceInterfaceEventMapping

Please note that the mapping of one sourceEvent to different targetEvents does **not** represent a *fan-out* of any kind.

It only means that the sourceEvent will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of ServiceInterfaces works in Figure A.5.

3.6.3 Service Interface Field Mapping

[TPS_MANI_01025]{DRAFT} **Semantics of ServiceInterfaceFieldMapping** ⌈ Meta-class ServiceInterfaceFieldMapping has the ability to map a ServiceInterface.field referenced in the role sourceField explicitly to another ServiceInterface.field referenced in the role targetField. ⌋*(RS_MANI_00017)*

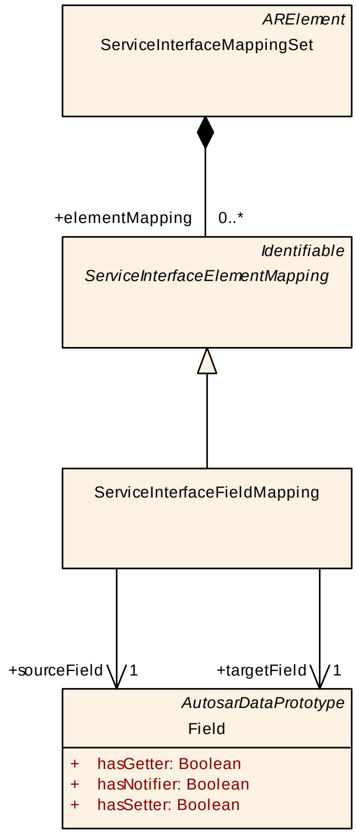


Figure 3.28: Modeling of the ServiceInterfaceFieldMapping

Class	**ServiceInterfaceFieldMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between fields of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceElementMapping*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
sourceField	Field	1	ref	Reference to a field that is contained in the source ServiceInterface. **Tags:** atp.Status=draft
targetField	Field	1	ref	Reference to a field that is contained in the composite ServiceInterface. **Tags:** atp.Status=draft

Table 3.37: ServiceInterfaceFieldMapping

The explicit mapping implemented by `ServiceInterfaceFieldMapping` does **not** require equal `shortName`s on both sides of the mapping.

It is also possible to map a given `field` of a given `ServiceInterface` multiple times in different roles to the `ServiceInterface` that aggregates the `targetField`, as exemplified by Figure 3.29.

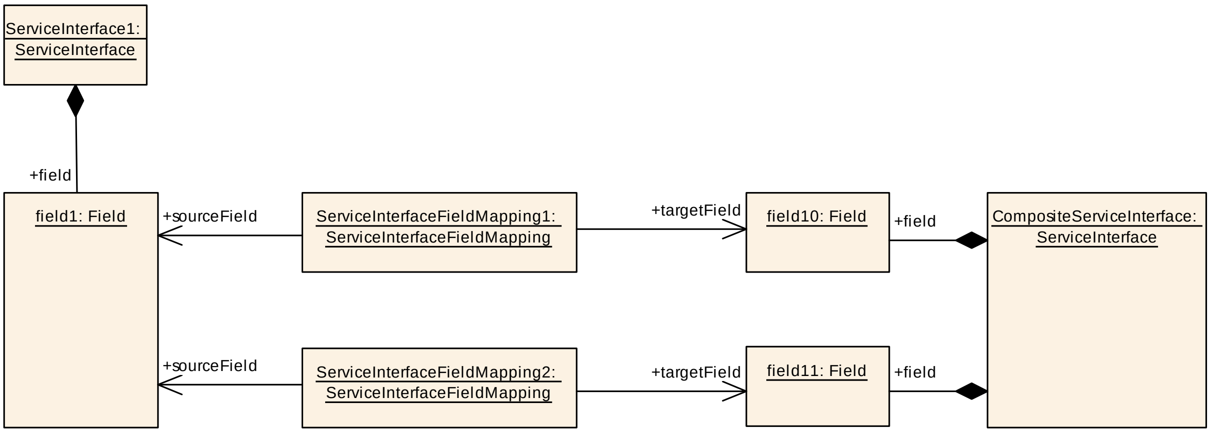


Figure 3.29: Example for the application of a ServiceInterfaceFieldMapping

Please note that the mapping of one sourceField to different targetFields does **not** represent a *fan-out* of any kind.

It only means that the sourceField will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of ServiceInterfaces works in Figure A.5.

3.6.4 Service Interface Method Mapping

[TPS_MANI_01026]{DRAFT} **Semantics of ServiceInterfaceMethodMapping**
⌈ Meta-class ServiceInterfaceMethodMapping has the ability to map a ServiceInterface.method referenced in the role sourceMethod explicitly to another ServiceInterface.method referenced in the role targetMethod. ⌋
(RS_MANI_00017)

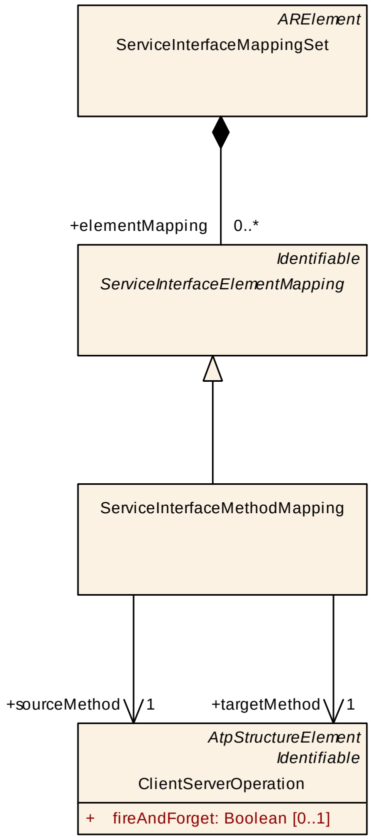


Figure 3.30: Modeling of the ServiceInterfaceMethodMapping

Class	**ServiceInterfaceMethodMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between methods of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceElementMapping*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
sourceMethod	ClientServerOperation	1	ref	Reference to a method that is contained in the source ServiceInterface. **Tags:** atp.Status=draft
targetMethod	ClientServerOperation	1	ref	Reference to a method that is contained in the composite ServiceInterface. **Tags:** atp.Status=draft

Table 3.38: ServiceInterfaceMethodMapping

The explicit mapping implemented by ServiceInterfaceMethodMapping does **not** require equal shortNames on both sides of the mapping.

It is also possible to map a given method of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetMethod, as exemplified by Figure 3.31.

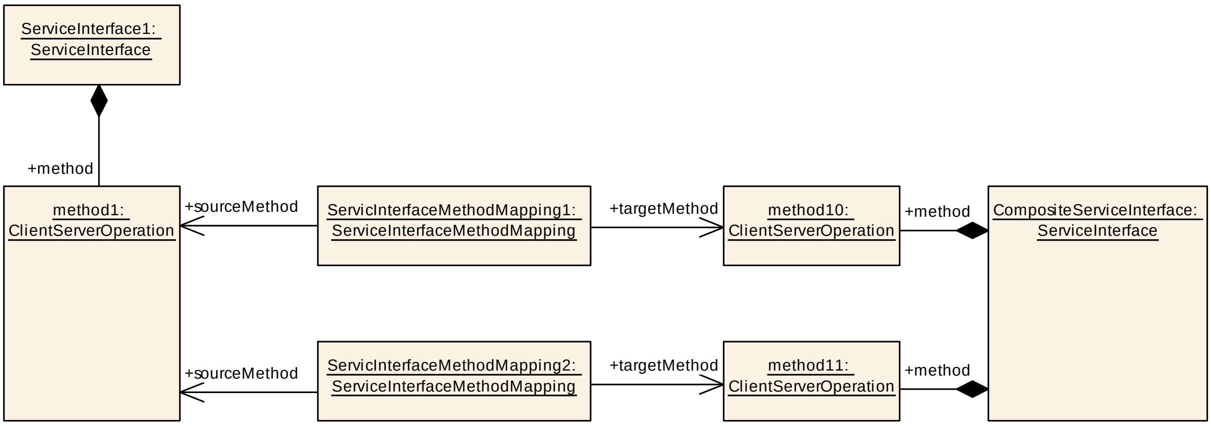


Figure 3.31: Example for the application of a ServiceInterfaceMethodMapping

Please note that the mapping of one sourceMethod to different targetMethods does **not** represent a *fan-out* of any kind.

It only means that the sourceMethod will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of ServiceInterfaces works in Figure A.5.

3.7 Service Needs

3.7.1 Overview

The vast majority of use cases for ServiceNeeds is applicable to the *AUTOSAR classic platform* and documented in the TPS Software Component Template [1].

However, as explained in section 4.1, there are also some case where ServiceNeeds can be successfully used also on the *AUTOSAR adaptive platform*.

For this purpose it is possible to reuse ServiceNeeds defined on the *AUTOSAR classic platform*. However, there are some use cases for the application of very specific subclasses of ServiceNeeds that are not available on the *AUTOSAR classic platform*.

The missing subclasses of meta-class ServiceNeeds are defined in this chapter.

3.7.2 Service Needs for Diagnostics

The introduction of the DiagnosticPortInterface (see section 3.11) and extensions to the DiagnosticMapping (as explained in section 4.1) for the purpose of implementing diagnostic communication on the *AUTOSAR adaptive platform* it is necessary to introduce further subclasses of ServiceNeeds.

Please note that this chapter contains a description of use cases for the diagnostic ServiceNeeds. The description looks very similar to the corresponding descriptions in the TPS Software Component Template.

The difference, however, is that the value of RoleBasedPortAssignment.role in the TPS Software Component Template describes the name of a PortInterface modeled on the M1 level while this chapter uses the names of meta-classes on M2.

[TPS_MANI_01256]{DRAFT} **AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticIndicatorInterface** ⌈ The aggregation of a DiagnosticIndicatorNeeds at a given SwcServiceDependency indicates a service use case where the application software implements a warning indicator.

ServiceNeeds kind DiagnosticIndicatorNeeds

RoleBasedPortAssignment valid roles:

- DiagnosticIndicatorInterface [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

⌋ *(RS_MANI_00061)*

Class	**DiagnosticIndicatorNeeds**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DiagnosticServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to implement an indicator. **Tags:** atp.Status=draft			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 3.39: DiagnosticIndicatorNeeds

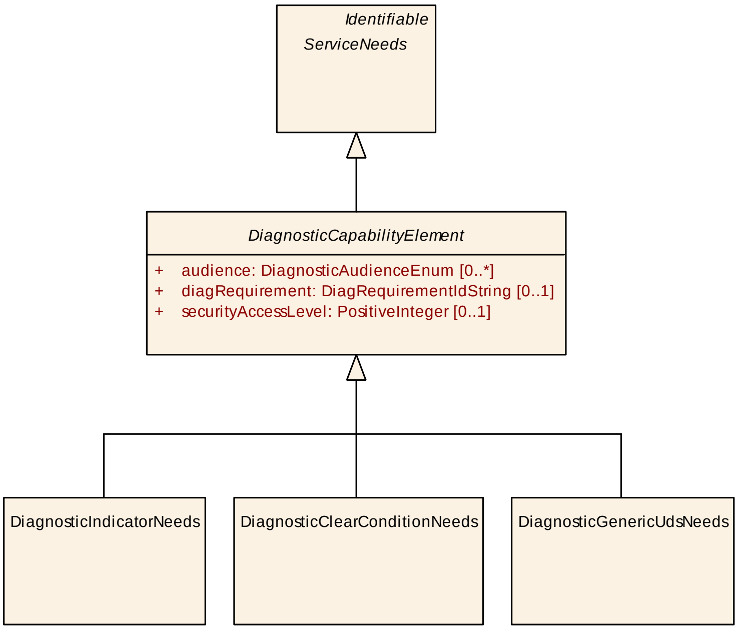


Figure 3.32: Modeling of diagnostic ServiceNeeds specifically for the *AUTOSAR adaptive platform*

[TPS_MANI_01257]{DRAFT} **AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticConditionInterface** ⌈ The aggregation of a DiagnosticClearConditionNeeds at a given SwcServiceDependency indicates a service use case where the application software implements a clear condition that can be queried by the Diagnostic Manager.

ServiceNeeds kind DiagnosticClearConditionNeeds

RoleBasedPortAssignment valid roles:

- DiagnosticConditionInterface [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

⌋(*RS_MANI_00061*)

Class	**DiagnosticClearConditionNeeds**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DiagnosticServiceNeeds
Note	This meta-class represents the needs of a software-component to provide the capability to set a clear condition. **Tags:** atp.Status=draft
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*

▽

△

Class	DiagnosticClearConditionNeeds			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 3.40: DiagnosticClearConditionNeeds

[TPS_MANI_01258]{DRAFT} **AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticGenericUdsInterface** ⌈ The aggregation of a DiagnosticGenericUdsNeeds at a given SwcServiceDependency indicates a service use case where the application software implements a generic handler of UDS services.

ServiceNeeds kind DiagnosticGenericUdsNeeds

RoleBasedPortAssignment valid roles:

- DiagnosticGenericUdsInterface [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

⌋*(RS_MANI_00061)*

Class	DiagnosticGenericUdsNeeds			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DiagnosticServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to process a generic UDS service. **Tags:** atp.Status=draft			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 3.41: DiagnosticGenericUdsNeeds

3.8 Persistency Interface

3.8.1 Overview

The *AUTOSAR adaptive platform* foresees a support for access to persistent data by e.g. application software.

There are some similarities to the communication model in terms of the usage of PortPrototypes.

In contrast to the configuration of communication, however, the modeling approach is much less detailed (i.e. instead of providing access to individual elements of a database an entire database is accessible on the level of PortPrototype).

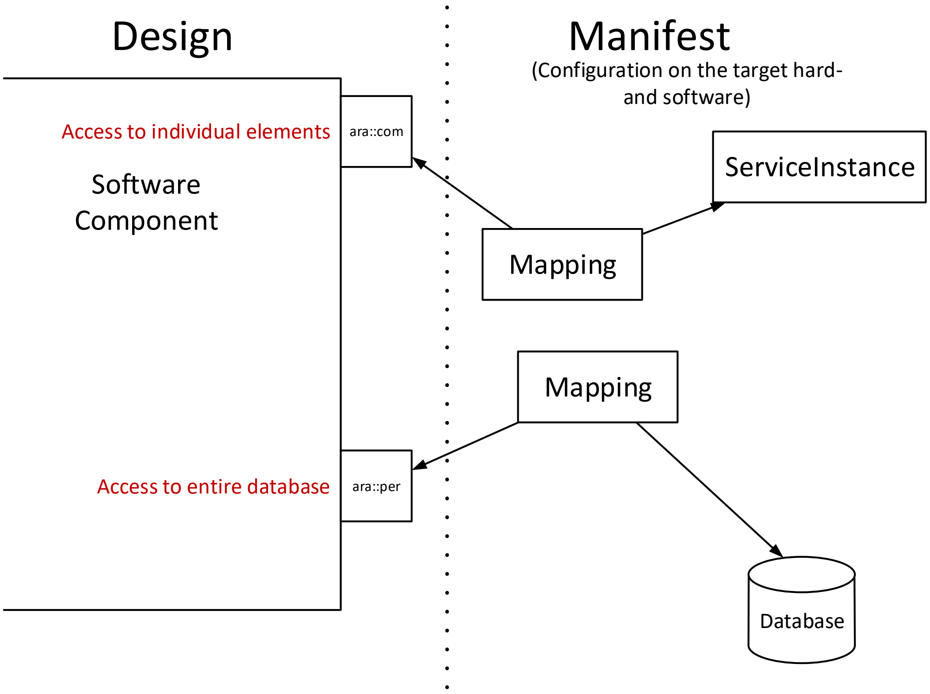


Figure 3.33: General approach for the modeling of persistency

The aspect of deployment for the configuration of persistent data is explained in Figure 3.33.

Please note that the AUTOSAR meta-model actually defines two separate meta-classes (for more details, please refer to Figure 3.34) for the different use cases of access to persistent data (i.e. PersistencyKeyValueDatabaseInterface) and access to files on the file system, or maybe an emulation of one (by means of PersistencyFileProxyInterface).

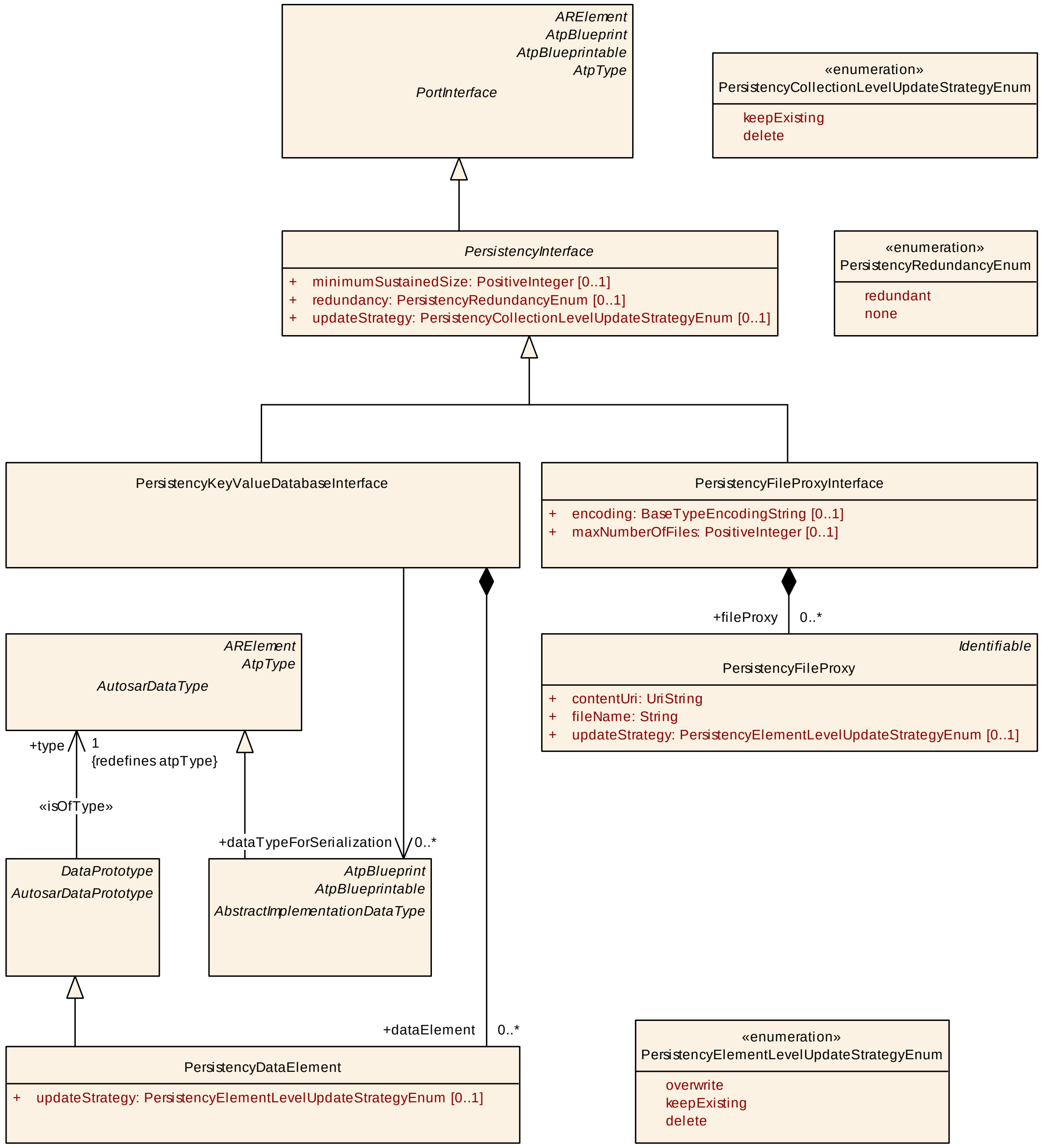


Figure 3.34: Specification of `PortInterface`s for persistency use cases

Abstract meta-class `PersistencyInterface` has been created as a means of categorization, i.e. it allows for easily referring to `PortInterface`s dedicated to persistency in general.

Class	*PersistencyInterface* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of persistency use cases. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	PersistencyFileProxyInterface, PersistencyKeyValueDatabaseInterface			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size required at design time for the enclosing Persistency Interface.
redundancy	PersistencyRedundancy Enum	0..1	attr	This attribute represents a requirement towards the redundancy of storage.
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyInterface as a whole.

Table 3.42: PersistencyInterface

[TPS_MANI_01204]{DRAFT} **Specification of redundancy of persistent data** ⌈ The attribute `PersistencyInterface.redundancy` can be taken to specify whether the respective key-value database or file shall store data redundantly from the perspective of the designer of the software-component. ⌋*(RS_MANI_00027)*

The details are left to an integrator who may also decide to overrule the value of `PersistencyInterface.redundancy` entirely if there is a use case for that.

Enumeration	PersistencyRedundancyEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class provides a way to specify in which way redundancy shall be applied on collection level. **Tags:** atp.Status=draft
Literal	**Description**
none	This value represents the requirement that redundancy measures are not applied on persistency collection level. **Tags:** atp.EnumerationValue=1
redundant	This value represents the requirement that redundancy measures are applied on persistency collection level. The nature of the redundant persistent storage is not further qualified and subject to integrator decisions. **Tags:** atp.EnumerationValue=0

Table 3.43: PersistencyRedundancyEnum

3.8.2 Persistency Key Value Database Interface

[TPS_MANI_01065]{DRAFT} **Purpose of `PersistencyKeyValueDatabaseInterface`** ⌈ The purpose of the `PersistencyKeyValueDatabaseInterface` is to support the persistent access to data in a key-value database. ⌋*(RS_MANI_00027)*

Class	**PersistencyKeyValueDatabaseInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for data. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyKeyValueDatabaseInterfaces			
Base	*ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PersistencyInterface, PortInterface, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataElement	PersistencyDataElement	*	aggr	This aggregation represents the collection of PersistencyDataElements in the context of the enclosing PersistencyKeyValueDatabaseInterface. **Tags:** atp.Status=draft
dataTypeForSerialization	AbstractImplementationDataType	*	ref	This reference identifies the AbstractImplementationDataTypes that shall be supported for storing in a key-value data base in addition to the types already referenced as PersistencyDataElement. **Tags:** atp.Status=draft

Table 3.44: PersistencyKeyValueDatabaseInterface

[TPS_MANI_01139]{DRAFT} **Semantics of `PersistencyKeyValueDatabaseInterface.updateStrategy`** ⌈The attribute `PersistencyKeyValueDatabaseInterface.updateStrategy` can be used to specify the strategy for updating the actual persistent elements used in the context of the `PersistencyKeyValueDatabase` that corresponds to `PersistencyKeyValueDatabaseInterface`.

This update strategy shall be applied to the `PersistencyKeyValueDatabaseInterface` as a whole except for the explicitly modeled `dataElement`s that define their own `updateStrategy`. ⌋*(RS_MANI_00027)*

The relation between a `PortPrototype` typed by a `PersistencyKeyValueDatabaseInterface` and the corresponding `PersistencyKeyValueDatabase` is described in section 8.2.2. The behavior of the software in terms of applying an update strategy is explained in detail in [9].

Enumeration	**PersistencyCollectionLevelUpdateStrategyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This enumeration provides possible values for the update strategy on interface/database level. **Tags:** atp.Status=draft
Literal	***Description***
delete	The update strategy is to delete all values on the level of the respective collection. **Tags:** atp.EnumerationValue=1
keepExisting	The update strategy is to keep the existing values on the level of the respective collection. **Tags:** atp.EnumerationValue=0

Table 3.45: PersistencyCollectionLevelUpdateStrategyEnum

[TPS_MANI_01135]{DRAFT} **Semantics of PersistencyKeyValueDatabaseInterface.dataTypeForSerialization** ⌈ The reference PersistencyKeyValueDatabaseInterface.dataTypeForSerialization can be taken to get information about data types for which a serialization algorithm has to be generated in order to support the persistent storage of objects of such data type. ⌋ *(RS_MANI_00027)*

In contrast to other kinds of PortInterfaces it is **not required** to define elements of a PersistencyKeyValueDatabaseInterface. If this is intended, however, the aggregation PersistencyKeyValueDatabaseInterface.dataElement shall be used for this purpose.

[TPS_MANI_01138]{DRAFT} **Semantics of PersistencyKeyValueDatabaseInterface.dataElement** ⌈ The definition of PersistencyKeyValueDatabaseInterface.dataElement supports the ability to generate transformer code as well as allow for a dedicated deployment of the dataElement to a given PersistencyKeyValueDatabase. ⌋ *(RS_MANI_00027)*

[TPS_MANI_01180]{DRAFT} **Collection of data types that requires serialization support** ⌈ The collection of data types that requires serialization support consists of

- AbstractImplementationDataTypes referenced in the role PersistencyKeyValueDatabaseInterface.dataTypeForSerialization
- either
  - AbstractImplementationDataTypes taken to type a PersistencyKeyValueDatabaseInterface.dataElement or
  - AbstractImplementationDataTypes mapped to ApplicationDataTypes taken to type a PersistencyKeyValueDatabaseInterface.dataElement by means of PortInterfaceToDataTypeMapping.dataTypeMappingSet that also refers to the enclosing PersistencyKeyValueDatabaseInterface.

⌋ *(RS_MANI_00027)*

[TPS_MANI_01194]{DRAFT} **Semantics of PersistencyKeyValueDatabaseInterface.minimumSustainedSize** ⌈ Attribute PersistencyKeyValueDatabaseInterface.minimumSustainedSize can be used for the definition of a minimum amount of storage that the PersistencyKeyValueDatabaseInterface will need to allocate from the application designer's point of view. ⌋ *(RS_MANI_00027)*

Class	**PersistencyDataElement**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface

▽

△

Class	PersistencyDataElement			
Note	This meta-class represents the ability to formally specify a piece of data that is subject to persistency in the context of the enclosing PersistencyKeyValueDatabaseInterface. PersistencyDataElement represents also a key of the deployed PersistencyKeyValueDatabase and provides an initial value. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	Type	Mul.	Kind	Note
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyDataElement.

Table 3.46: PersistencyDataElement

[TPS_MANI_01140]{DRAFT} **Semantics of PersistencyDataElement.updateStrategy** ⌈ The attribute PersistencyDataElement.updateStrategy can be used to specify the strategy for updating the actual persistent element that corresponds to PersistencyDataElement. ⌋*(RS_MANI_00027)*

The relation between a PersistencyDataElement and the corresponding PersistencyKeyValuePair in the scope of a PersistencyKeyValueDatabase is described in section 8.2.2. The behavior of the software in terms of applying an update strategy for specific persistent elements is explained in detail in [9].

Enumeration	PersistencyElementLevelUpdateStrategyEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This enumeration provides possible values for the update strategy on element level. **Tags:** atp.Status=draft
Literal	Description
delete	The update strategy is to delete the value of the respective data item. **Tags:** atp.EnumerationValue=2
keepExisting	The update strategy is to keep the existing value of the respective data item. **Tags:** atp.EnumerationValue=1
overwrite	The update strategy is to overwrite the respective data item. **Tags:** atp.EnumerationValue=0

Table 3.47: PersistencyElementLevelUpdateStrategyEnum

Please note that a PersistencyDataElement can be typed by either an ApplicationDataType or else a CppImplementationDataType.

3.8.3 Persistency File Proxy Interface

[TPS_MANI_01067]{DRAFT} **Purpose of PersistencyFileProxyInterface** ⌈ The purpose of meta-class PersistencyFileProxyInterface is to support access to an abstract representation of files. ⌋*(RS_MANI_00027)*

[constr_1524]{DRAFT} **Standardized values of PersistencyFileProxyInterface.category** ⌈ The values of PersistencyFileProxyInterface.category shall be taken to further qualify the nature of the accessed files. The following values are standardized:

- TEXT_FILE
- BINARY_FILE

⌋*()*

[TPS_MANI_01068]{DRAFT} **Semantics of PersistencyFileProxyInterface.maxNumberOfFiles** ⌈ Any PortPrototype typed by a PersistencyFileProxyInterface has the ability to access a number of files.

The upper bound of the number of files represented by a given PortPrototype typed by a PersistencyFileProxyInterface can be configured using the attribute PersistencyFileProxyInterface.maxNumberOfFiles.

The value of attribute PersistencyFileProxyInterface.maxNumberOfFiles **includes** the explicitly modeled PersistencyFileProxyInterface.fileProxy.
⌋*(RS_MANI_00027)*

Please note that the existence of the PersistencyFileProxyInterface does not violate the restrictions set by the POSIX subset PSE51 defined in IEEE1003.13 [10].

Class	**PersistencyFileProxyInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for files. **Tags:** atp.Status=draft atp.recommendedPackage=PersistencyFileProxyInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
encoding	BaseTypeEncodingString	0..1	attr	This attribute supports the definition of an encoding of the corresponding physical files. The possible values of this attribute may be partially standardized by AUTOSAR. But it is also possible to extend the set of values in a custom way (provided that the custom values use a notation that ensures the absence of clashes with further extensions of the standardized values, e.g. by using a company-specific prefix).
fileProxy	PersistencyFileProxy	*	aggr	This aggregation represents the collection of PersistencyFileProxys in the context of the enclosing PersistencyFileProxyInterface. **Tags:** atp.Status=draft

▽

△

Class	PersistencyFileProxyInterface			
maxNumberOf Files	PositiveInteger	0..1	attr	This attribute represents the definition of an upper bound for the handling of files at run-time in the context of the enclosing PersistencyFileProxyInterface.

Table 3.48: PersistencyFileProxyInterface

A PortPrototype typed by a PersistencyFileProxyInterface allows for abstracting the actual calls to the operating system away from the scope of the application software and into the modules of the *AUTOSAR adaptive platform*.

[TPS_MANI_01141]{DRAFT} **Semantics of PersistencyFileProxyInterface.updateStrategy** ⌈ The attribute PersistencyFileProxyInterface.updateStrategy can be used to specify the strategy for updating the actual persistent files used in the context of the PersistencyFileArray that corresponds to PersistencyFileProxyInterface.

This update strategy shall be applied to the PersistencyFileProxyInterface as a whole except for the explicitly modeled fileProxys that define their own updateStrategy. ⌋*(RS_MANI_00027)*

The relation between a PortPrototype typed by a PersistencyFileProxyInterface and the corresponding PersistencyFileArray is described in section 8.2.2. The behavior of the software in terms of applying an update strategy is explained in detail in [9].

[TPS_MANI_01195]{DRAFT} **Semantics of PersistencyFileProxyInterface.minimumSustainedSize** ⌈ Attribute PersistencyFileProxyInterface.minimumSustainedSize can be used for the definition of a minimum amount of storage that the PersistencyFileProxyInterface will need to allocate from the application designer's point of view.

It is the responsibility of the underlying platform to make sure that this minimum amount of storage is available at any time. ⌋*(RS_MANI_00027)*

[TPS_MANI_01142]{DRAFT} **Semantics of PersistencyFileProxy** ⌈ By aggregating PersistencyFileProxy in the role fileProxy it is possible to explicitly model files (and some of their properties) accessible to the application software within the context of a PersistencyFileProxyInterface. ⌋*(RS_MANI_00027)*

Class	PersistencyFileProxy
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This meta-class has the ability to represent a file at design time such that it is possible to configure the behavior for accessing the represented file at run-time. **Tags:** atp.Status=draft
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	PersistencyFileProxy			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
contentUri	UriString	1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	1	attr	This attribute holds filename part of the storage location for the PersistencyFileProxy, e.g. file on the file system.
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyFileProxy.

Table 3.49: PersistencyFileProxy

[TPS_MANI_01143]{DRAFT} **Semantics of PersistencyFileProxy.updateStrategy** ⌈ The attribute PersistencyFileProxy.updateStrategy can be used to specify the strategy for updating the actual persistent file that corresponds to PersistencyFileProxy. ⌋*(RS_MANI_00027)*

The behavior of the software in terms of applying an update strategy for specific persistent files is explained in detail in [9].

[constr_1581]{DRAFT} **Value of fileProxy.fileName** ⌈ Within the scope of any given PersistencyFileProxyInterface, the value of all fileProxy.fileName shall be unique. ⌋*()*

3.9 Time Synchronization Interface

Time Synchronization functional cluster within the Adaptive Platform is responsible to provide various Time Bases for the application to read from or to write to.

In order to interface with the Time Synchronization foundation software an application developer needs to declare which kind of Time Base this application will interact with.

The interface towards the Time Synchronization follows the generic pattern of PortPrototypes and PortInterfaces which are applied to many use-cases concerning the interaction of application software with platform software.

In contrast to the service based communication, the modeling of platform software interaction using PortPrototypes and PortInterfaces is less detailed. The PortPrototype is a placeholder for the interaction with platform software, it does not model the actually used APIs available for the interaction. The APIs to be used are formally specified in the platform software SWS document, i.e. SWS_TimeSync [11].

[TPS_MANI_03535]{DRAFT} **Definition of Time Synchronization interaction** ⌈ The meta-class TimeSynchronizationInterface together with its sub classes are used to define the interaction of the application software with a Time Synchronization Time Base. ⌋*(RS_MANI_00040)*

For more information, pleaser refer to Figure 3.35.

By defining an RPortPrototype which is typed by one of the TimeSynchronizationInterface sub classes the application indicates that it will access a specific Time Base.

[TPS_MANI_03549]{DRAFT} **Usage of RPortPrototype for the interaction with Time Synchronization** ⌈ When defining a PortPrototype typed by one of the subclasses of TimeSynchronizationInterface an RPortPrototype shall be used. ⌋*(RS_MANI_00040)*

The application software takes the active role in the interaction with foundation platform software thus a RPortPrototype is used to represent this interaction from the application software point of view.

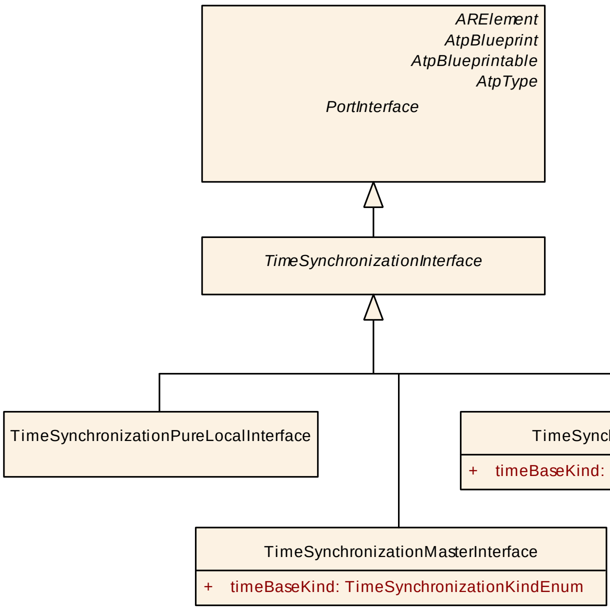


Figure 3.35: Modeling of Time Synch Interfaces

[TPS_MANI_03536]{DRAFT} **Time Synchronization interaction in a master role** ⌈ The meta-class TimeSynchronizationMasterInterface is used to indicate the intended interaction with a synchronized global Time Base in a *master* role. ⌋ *(RS_MANI_00040)*

When interacting with a synchronized global Time Base in a *master* role the application is able to *set* (and *get*) the value of the synchronized global Time Base which is then propagated to the time value on the network.

[TPS_MANI_03537]{DRAFT} **Time Synchronization interaction in a slave role** ⌈ The meta-class TimeSynchronizationSlaveInterface is used to indicate the intended interaction with a synchronized global Time Base in a *slave* role. ⌋ *(RS_MANI_00040)*

When interacting with a synchronized global Time Base in a *slave* role the application is able to only *get* the value of the synchronized global Time Base which is synchronized from a time value coming from the network.

[TPS_MANI_03551]{DRAFT} **Definition of Time Base kind** ⌈ The attributes TimeSynchronizationMasterInterface.timeBaseKind and TimeSynchronizationSlaveInterface.timeBaseKind define whether the Time Base shall be a synchronized or a offset Time Base. ⌋*(RS_MANI_00040)*

[TPS_MANI_03538]{DRAFT} **Time Synchronization interaction with a local Time Base** ⌈ The meta-class TimeSynchronizationPureLocalInterface is used to indicate the intended interaction with a pure local Time Base. ⌋*(RS_MANI_00040)*

When interacting with a pure local Time Base the application is able to *set* and *get* the value of the Time Base, but the value is considered local to the Machine.

Class	**TimeSynchronizationMasterInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Master. **Tags:** atp.Status=draft atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *TimeSynchronizationInterface*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
timeBaseKind	TimeSynchronizationKindEnum	1	attr	Defines which kind of time base is requested at this interface. **Tags:** atp.Status=draft

Table 3.50: TimeSynchronizationMasterInterface

Class	**TimeSynchronizationSlaveInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Slave. **Tags:** atp.Status=draft atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *TimeSynchronizationInterface*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
timeBaseKind	TimeSynchronizationKindEnum	1	attr	Defines which kind of time base is requested at this interface. **Tags:** atp.Status=draft

Table 3.51: TimeSynchronizationSlaveInterface

Class	**TimeSynchronizationPureLocalInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface

▽

△

Class	**TimeSynchronizationPureLocalInterface**			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Pule Local Time Base. **Tags:** atp.Status=draft atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable, Time SynchronizationInterface*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 3.52: TimeSynchronizationPureLocalInterface

Enumeration	**TimeSynchronizationKindEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	Defines the possible kinds of TimeSynchronizationInterfaces. **Tags:** atp.Status=draft
Literal	*Description*
offset	Defines that the requested time base shall be an offset time based. **Tags:** atp.EnumerationValue=1
synchronized	Defines that the requested time base shall be a synchronized time based. **Tags:** atp.EnumerationValue=0

Table 3.53: TimeSynchronizationKindEnum

In the example in figure 3.36 the interaction of one Application with several time sync aspects are illustrated.

The interaction approach is that for each meta-class of `TimeSynchronizationInterface` used in an application's `RPortPrototype` a corresponding *TimeBaseResourceProxy* is generated and the application developer gains access to the time synchronization kind by use of this *TimeBaseResourceProxy*.

In the application code the *TimeBaseResourceProxy* is used to initiate a *find* functionality using *InstanceIdentifier* representing the `RPortPrototype` as defined in [12] and described in [13].

During application deployment those `RPortPrototype`s are mapped to actual *Time Bases* in the Time Sync Management (see figure 8.17).

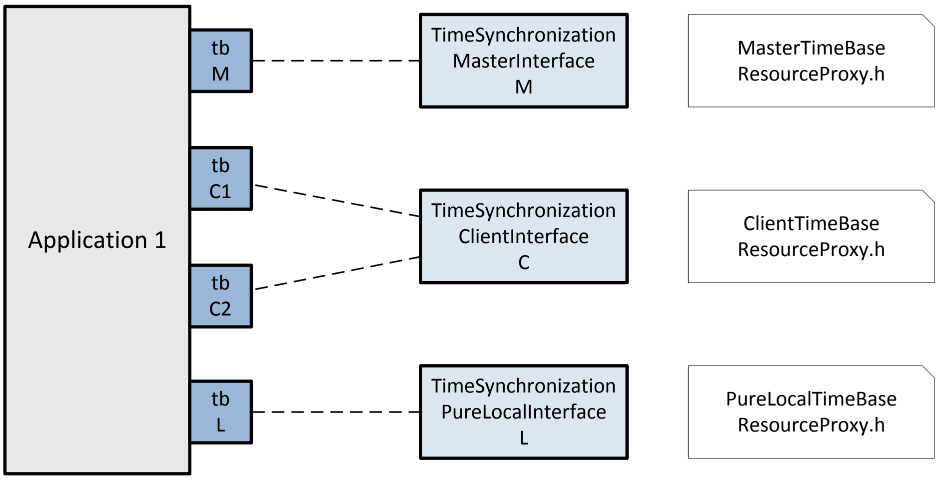


Figure 3.36: Example Application and Time Sync interaction

3.10 Platform Health Management Interface

3.10.1 Overview

Platform Health Management functional cluster within the Adaptive Platform is responsible to supervise the execution of applications, monitor their status, provide rule-based evaluation of the status, and triggering of respective actions.

In order to interface with the Platform Health Management foundation software an application developer needs to declare which supervisions and status information is provided by the application software and shall be observed by the Platform Health Management.

The interface towards the Platform Health Management follows the generic pattern of PortPrototypes and PortInterfaces which are applied to many use-cases concerning the interaction of application software with platform software.

In contrast to the service based communication, the modeling of platform software interaction using PortPrototypes and PortInterfaces is less detailed. The PortPrototype is a placeholder for the interaction with platform software, it does not model the actually used APIs available for the interaction. The APIs to be used are formally specified in the platform software SWS document, i.e. SWS_HealthManagement [14].

3.10.2 Supervised Entities and Checkpoints

The interaction of supervision with the Platform Health Management is defined by PhmSupervisedEntityInterface and PhmCheckpoints.

[TPS_MANI_03500]{DRAFT} **Definition of Platform Health Management Supervision and Checkpoints** ⌈ The meta-class PhmSupervisedEntityInterface together with the aggregated PhmCheckpoint are used to define the interaction of one Supervised Entity with the Platform Health Management supervision. ⌋ *(RS_MANI_00032)*

By defining an RPortPrototype which is typed by the PhmSupervisedEntityInterface the application indicates that it wants to report the checkpoints of this PhmSupervisedEntityInterface.

[TPS_MANI_03550]{DRAFT} **Usage of RPortPrototype for the interaction with Platform Health Management** ⌈ When defining a PortPrototype typed by one of the sub-classes of PlatformHealthManagementInterface an RPortPrototype shall be used. ⌋*(RS_MANI_00040)*

The application software takes the active role in the interaction with foundation platform software thus a RPortPrototype is used to represent this interaction from the application software point of view. The *SupervisedEntity* instance is constructed using the InstanceSpecifier of the respective RPortPrototype.

The application code then calls the *ReportCheckpoint* API (defined in [14]) of the *SupervisedEntity* (which has been constructed in the context of the respective RPortPrototype typed by the PhmSupervisedEntityInterface) in order to notify the Platform Health Management that a specific PhmCheckpoint has been reached in the program flow.

[constr_3530]{DRAFT} **Mandatory definition of checkpointId** ⌈ The checkpointId shall be defined for every PhmCheckpoint element. ⌋*()*

The checkpointId is used during the call to the *ReportCheckpoint* API as a representation of the PhmCheckpoint.

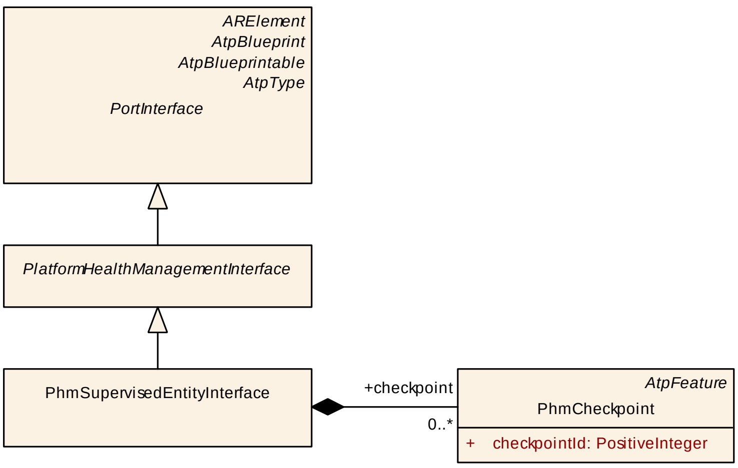


Figure 3.37: Modeling of Supervised Entities and Checkpoints

If the application wants to query the status of a Supervised Entity monitored by the Platform Health Management then the application code calls the *GetLocalSupervisionStatus* API (defined in [14]) of the *SupervisedEntity* (which has been constructed in the context of the respective RPortPrototype typed by the PhmSupervisedEntity-Interface).

Note that from the application design point of view there are no relations defined between the checkpoints (as to indicate a specific observed order in reporting). The possible transitions between the checkpoints and their timing aspects are defined in the context of the PlatformHealthManagementContribution and described in chapter 8.3.2.

Class	**PhmSupervisedEntityInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Platform Health Management Supervised Entity. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformHealthManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *Port Interface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
checkpoint	PhmCheckpoint	*	aggr	Defines the set of checkpoints which can be reported on this supervised entity. **Tags:** atp.Status=draft

Table 3.54: PhmSupervisedEntityInterface

Class	**PhmCheckpoint**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a checkpoint for interaction with the Platform Health Management Supervised Entity. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
checkpointId	PositiveInteger	1	attr	Defines the numeric value which is used to indicate the reporting of this Checkpoint to the Phm. **Tags:** atp.Status=draft

Table 3.55: PhmCheckpoint

3.10.3 Health Channels

The interaction of Health Channels with the Platform Health Management is defined by PhmHealthChannelInterface and PhmHealthChannelStatus states.

[TPS_MANI_03534]{DRAFT} **Definition of Platform Health Management Health Channel** ⌈ The meta-class PhmHealthChannelInterface together with the aggregated PhmHealthChannelStatus are used to define the interaction of one Health Channel with the Platform Health Management. ⌋*(RS_MANI_00032)*

By defining a RPortPrototype which is typed by the PhmHealthChannelInterface (see [TPS_MANI_03550]) the application indicates that it wants to report the status of this PhmHealthChannelInterface.

The application software takes the active role in the interaction with foundation platform software thus a RPortPrototype is used to represent this interaction from the application software point of view. The *HealthChannel* instance is constructed using the InstanceSpecifier of the respective RPortPrototype.

The application code then calls the *ReportHealthStatus* API (defined in [14]) of the *HealthChannel* (which has been constructed in the context of the respective RPortPrototype typed by the PhmHealthChannelInterface) in order to notify the Platform Health Management that the Health Channel defined by the RPortPrototype has changed its status.

[constr_3532]{DRAFT} **Mandatory definition of statusId** ⌈ The statusId shall be defined for every PhmHealthChannelStatus element. ⌋*()*

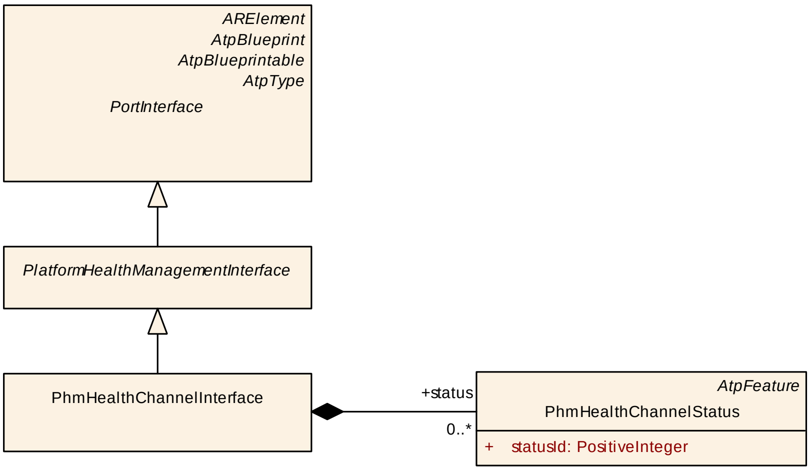


Figure 3.38: Modeling of Health Channel

Class	**PhmHealthChannelInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Platform Health Management Health Channel. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformHealthManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *Port Interface*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
status	PhmHealthChannel Status	*	aggr	Defines the possible set of status information available to the health channel. **Tags:** atp.Status=draft

Table 3.56: PhmHealthChannelInterface

Class	PhmHealthChannelStatus			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	The PhmHealthChannelStatus specifies one possible status of the health channel. **Tags:** atp.Status=draft			
Base	*ARObject, AtpFeature, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
statusId	PositiveInteger	1	attr	Defines the numeric value which is used to indicate the indication of this status the Phm. **Tags:** atp.Status=draft

Table 3.57: PhmHealthChannelStatus

3.11 Diagnostic Interface

3.11.1 Overview

On the *AUTOSAR adaptive platform*, dedicated `PortInterface`s are defined for the interaction of application-layer software with the AUTOSAR `Diagnostic Manager`.

In contrast to the conventions on the AUTOSAR classic Platform, these `PortInterface`s and, by extension, the standardized `ara::diag` API **are only used on the application side** of this communication relation.

The interfaces on the side of the AUTOSAR `Diagnostic Manager` (and thus the part of the implementation of the `PortPrototype` that faces the AUTOSAR `Diagnostic Manager`) are **entirely proprietary**. This aspect is depicted in Figure 3.39.

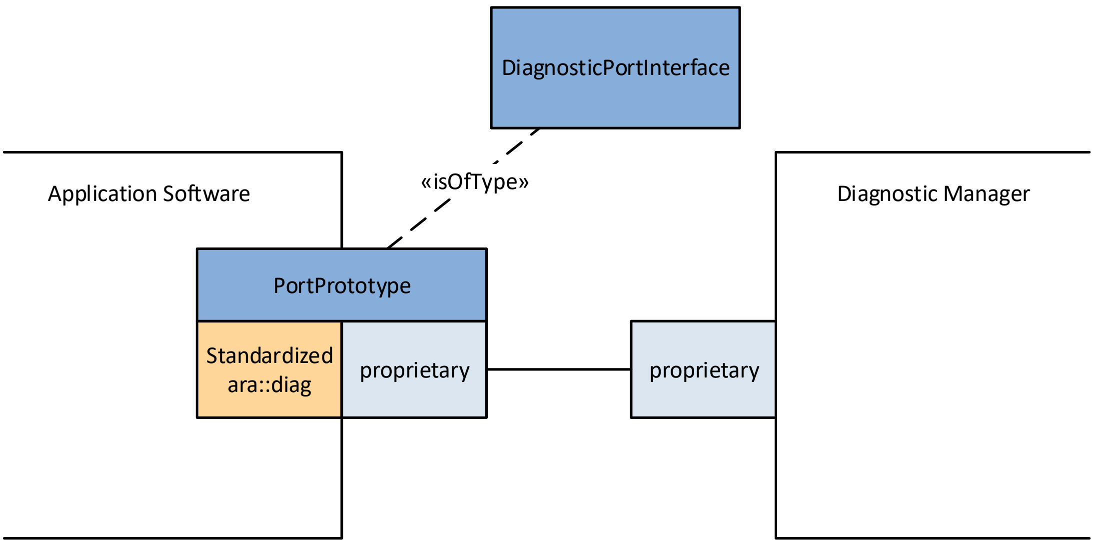


Figure 3.39: Standardized vs. proprietary parts in the implementation of `ara::diag`

This arrangement tries to provide the application programmer with the simplest possible API from the application's point of view. At the same time it hides a lot of the complexity of the interaction between application and `Diagnostic Manager` behind a solid abstraction layer.

[TPS_MANI_01242]{DRAFT} **`PortInterface`s used for communication with the AUTOSAR `Diagnostic Manager`** ⌈ All `PortInterface`s used for this purpose ar derived from the abstract meta-class `DiagnosticPortInterface`. A `DiagnosticPortInterface` does not implement a service-oriented communication pattern, in particular there is no explicit service discovery on the API level involved. ⌋ *(RS_MANI_00061)*

The specializations of `DiagnosticPortInterface` cover the various aspects of diagnostic communication, e.g. the implementation of diagnostic routines, the reporting of diagnostic events or the access to a Diagnostic Data Identifier (DID).

Class	***DiagnosticPortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*DiagnosticAbstractDataIdentifierInterface*, *DiagnosticAbstractRoutineInterface*, DiagnosticConditionInterface, DiagnosticDTCInformationInterface, DiagnosticDoIPGroupIdentificationInterface, DiagnosticDoIPPowerModeInterface, DiagnosticDownloadInterface, DiagnosticEventInterface, DiagnosticGenericUdsInterface, DiagnosticIndicatorInterface, DiagnosticMonitorInterface, DiagnosticOperationCycleInterface, DiagnosticSecurityLevelInterface, DiagnosticServiceValidationInterface, DiagnosticUploadInterface			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.58: DiagnosticPortInterface

Figure 3.40 depicts all meta-classes that directly inherit from `DiagnosticPortInterface`.

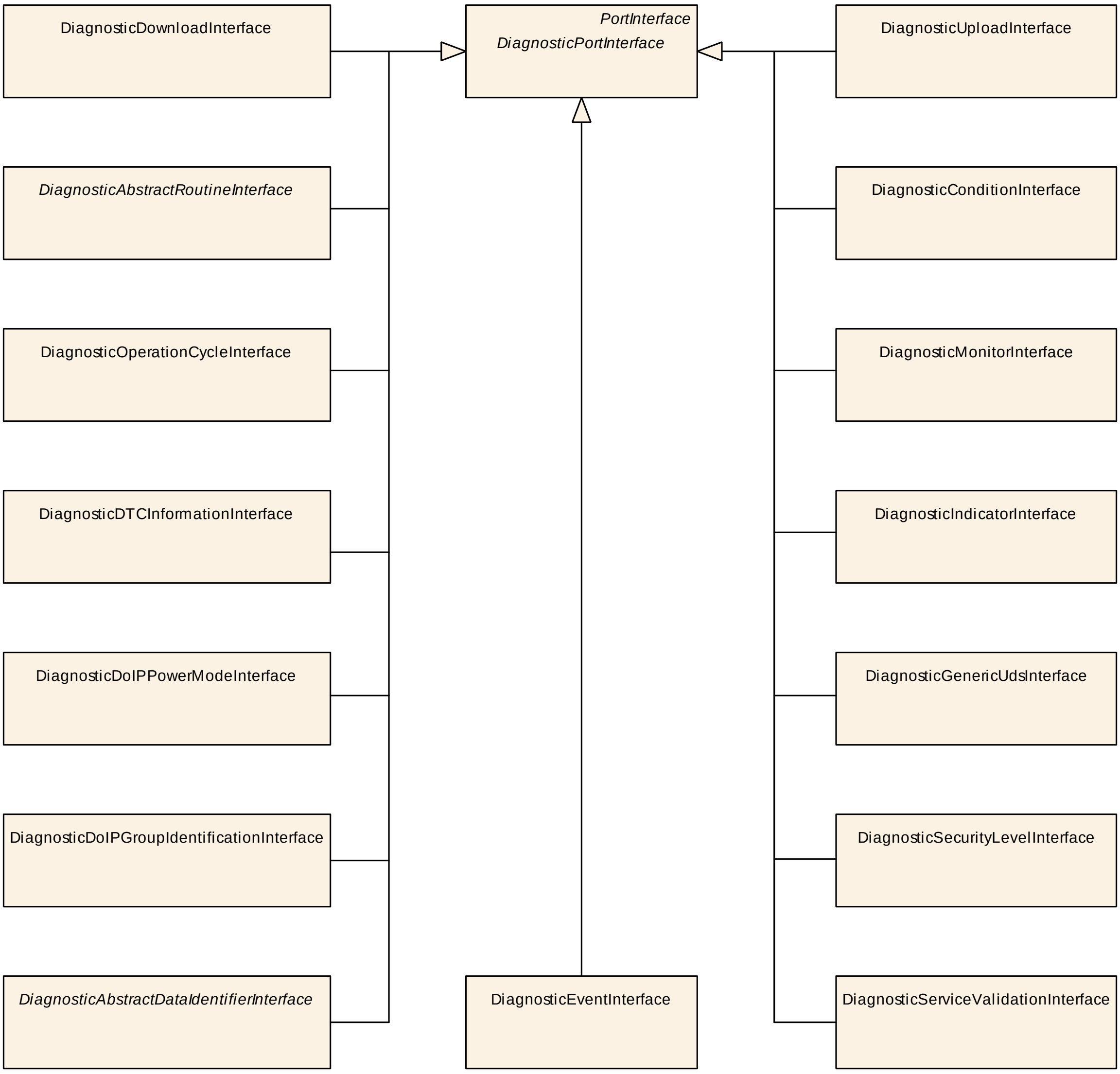


Figure 3.40: Modeling of PortInterfaces for diagnostic purposes

3.11.2 Diagnostic Routine Interface

The convention for the creation of diagnostic routines is to establish at most three methods for each diagnostic routine:

- Start the execution of the routine.
- Stop the execution of the routine.
- Request the results of the routine's execution.

In response to this convention the DiagnosticRoutineInterface is modeled to aggregate ClientServerOperation in three dedicated roles: start, stop, and requestResult.

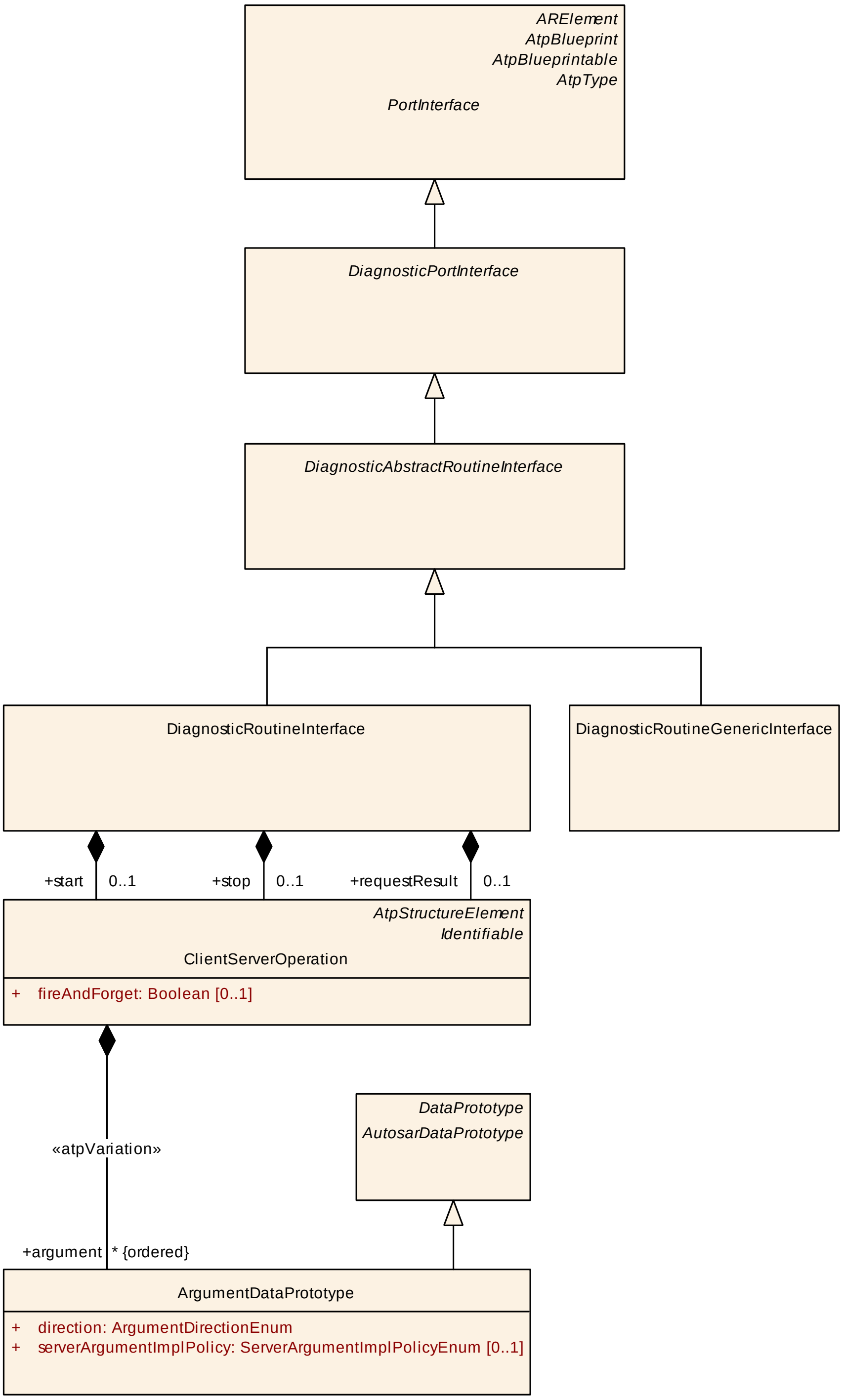


Figure 3.41: Modeling of DiagnosticRoutineInterface

[constr_1696]{DRAFT} **ClientServerOperation aggregated by DiagnosticRoutineInterface** ⌈ Any ClientServerOperation aggregated by a DiagnosticRoutineInterface shall not define the following attributes:

- fireAndForget
- possibleApError

- possibleApErrorSet

⌋()

Class	DiagnosticRoutineInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a routine-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticAbstractRoutineInterface, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
requestResult	ClientServerOperation	0..1	aggr	This represents the request result method of the diagnostic routine. **Tags:** atp.Status=draft
start	ClientServerOperation	0..1	aggr	This represents the start method of the diagnostic routine. **Tags:** atp.Status=draft
stop	ClientServerOperation	0..1	aggr	This represents the stop method of the diagnostic routine. **Tags:** atp.Status=draft

Table 3.59: DiagnosticRoutineInterface

The arguments to the diagnostic routine shall be modeled as the arguments of the respective ClientServerOperations aggregated in the roles start, stop, and requestResult.

In addition to the modeling of "typed" diagnostic routines using the DiagnosticRoutineInterface it is possible to use the DiagnosticRoutineGenericInterface to define a diagnostic routine for which no further formalization is provided.

Class	DiagnosticRoutineGenericInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic Routine-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticAbstractRoutineInterface, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.60: DiagnosticRoutineGenericInterface

This means that implicitly there are still up to three methods defined for the already mentioned roles of a diagnostic routine.

However, the methods inside the context of such a generic diagnostic routine would always use plain byte arrays as the arguments and therefore a formalization within the AUTOSAR meta-model does not make sense any longer.

Meta-class DiagnosticAbstractRoutineInterface serves as the abstract base class to all routine-related DiagnosticPortInterfaces on the *AUTOSAR adaptive platform*.

Class	***DiagnosticAbstractRoutineInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class serves as the abstract base class of PortInterfaces dedicated to routine execution on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticRoutineGenericInterface, DiagnosticRoutineInterface			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.61: DiagnosticAbstractRoutineInterface

3.11.3 Interface to Data Identifier and Element of Data Identifier

The ability to access diagnostic-relevant **data** in the application software is formalized in another abstract sub-class of DiagnosticPortInterface: DiagnosticAbstractDataIdentifierInterface.

Meta-class DiagnosticAbstractDataIdentifierInterface, in turn, defines three concrete subclasses that represent the concrete abilities to access diagnostic-related data in the application software.

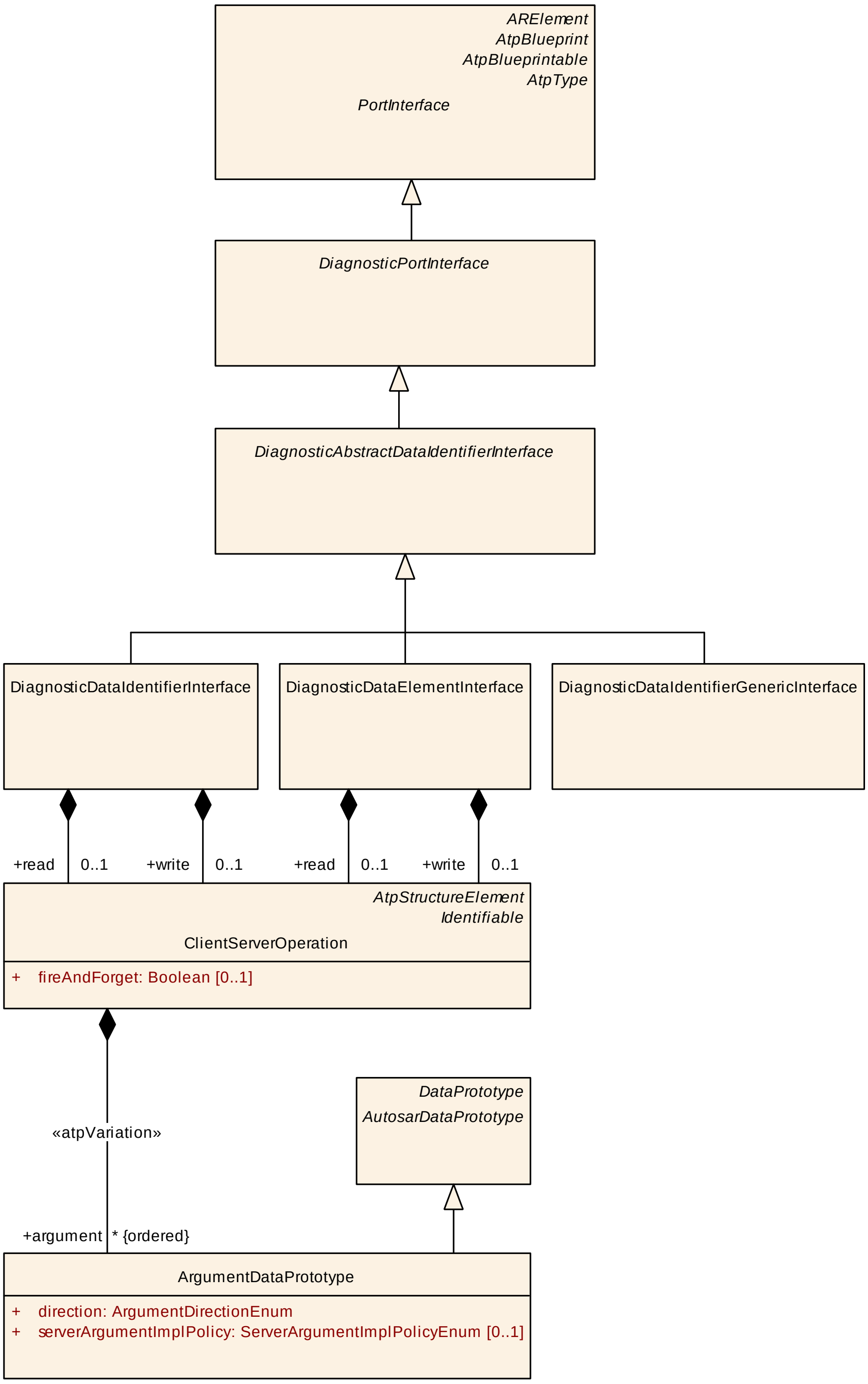


Figure 3.42: Modeling of DiagnosticDataIdentifierInterface

[TPS_MANI_01243]{DRAFT} **Semantics of DiagnosticDataIdentifierInterface** ⌈DiagnosticDataIdentifierInterface is used to access the content of an entire DID at once.

For this purpose up to two ClientServerOperations are aggregated in the roles read and write, depending on the concrete use case for a specific DiagnosticDataIdentifierInterface. ⌋*(RS_MANI_00061)*

Class	DiagnosticDataIdentifierInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of a diagnostic data identifier. **Tags:** atp.Status=draft
write	ClientServerOperation	0..1	aggr	This represents the method to write the contents of a diagnostic data identifier. **Tags:** atp.Status=draft

Table 3.62: DiagnosticDataIdentifierInterface

[TPS_MANI_01244]{DRAFT} **Semantics of DiagnosticDataElementInterface** ⌈ DiagnosticDataElementInterface is used to access the content of an element within a given DID.

For this purpose up to two ClientServerOperations are aggregated in the roles read and write, depending on the concrete use case for a specific DiagnosticDataElementInterface. ⌋*()*

Class	DiagnosticDataElementInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a element-of-DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of an element of a diagnostic data identifier. **Tags:** atp.Status=draft
write	ClientServerOperation	0..1	aggr	This represents the method to write the content of an element of a diagnostic data identifier. **Tags:** atp.Status=draft

Table 3.63: DiagnosticDataElementInterface

[TPS_MANI_01245]{DRAFT} **Semantics of DiagnosticDataIdentifierGenericInterface** ⌈ DiagnosticDataIdentifierInterface is used to access the content of an entire DID at once.

For this purpose methods will be defined with a read and write semantics, but these methods will always only provide arguments that are byte-arrays.

Therefore, a further formalization of these methods for reading and writing data within the context of the AUTOSAR meta-model does not make sense and is therefore omitted. ⌋*(RS_MANI_00061)*

Class	DiagnosticDataIdentifierGenericInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.64: DiagnosticDataIdentifierGenericInterface

Please note that it is necessary to put some restrictions on the argument **unless** a given DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface aggregates **only one** ClientServerOperation in **either** the role read **or** write.

[constr_1697]{DRAFT} **Restriction for ClientServerOperation aggregated by a DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface** ⌈ If meta-classes DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface aggregate two ClientServerOperations then

- The two ClientServerOperations shall have the same number of arguments.
- The arguments on the n^{th} position in the collection of arguments shall have identical properties, except the direction. In particular, the following conditions shall be fulfilled with respect to attribute direction:
  - Any ArgumentDataPrototype aggregated by a ClientServerOperation that is itself aggregated in either the role DiagnosticDataIdentifierInterface.read or DiagnosticDataElementInterface.read shall set attribute direction to out.
  - Any ArgumentDataPrototype aggregated by a ClientServerOperation that is itself aggregated in either the role DiagnosticDataIdentifierInterface.write or DiagnosticDataElementInterface.write shall set attribute direction to in.

⌋*()*

3.11.4 Interface to diagnostic Events

AUTOSAR defines several subclasses of DiagnosticPortInterface that are dedicated to the handling of diagnostic events.

[TPS_MANI_01246]{DRAFT} **Semantics of DiagnosticMonitorInterface** ⌈ Meta-class DiagnosticMonitorInterface represents the ability to report diagnostic events to the AUTOSAR Diagnostic Manager. ⌋*(RS_MANI_00061)*

Class	**DiagnosticMonitorInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.65: DiagnosticMonitorInterface

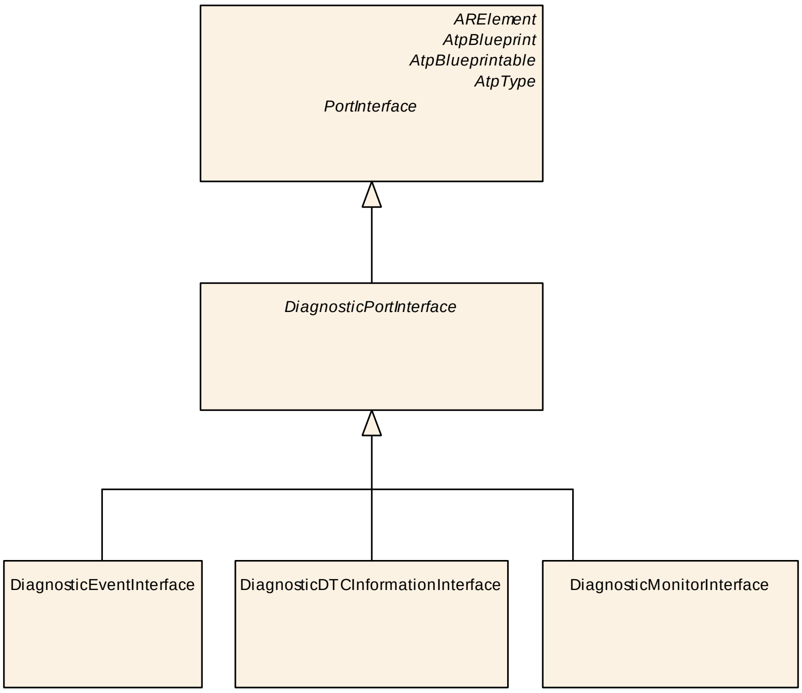


Figure 3.43: Modeling of DiagnosticEventInterface

[TPS_MANI_01247]{DRAFT} **Semantics of DiagnosticDTCInformationInterface** ⌈ Meta-class DiagnosticDTCInformationInterface represents the ability to retrieve information about a given diagnostic trouble code. ⌋ *(RS_MANI_00061)*

Class	DiagnosticDTCInformationInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of DTCs on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.66: DiagnosticDTCInformationInterface

[TPS_MANI_01248]{DRAFT} **Semantics of** `DiagnosticEventInterface` ⌈ Meta-class `DiagnosticEventInterface` represents the ability to retrieve information about a given diagnostic event. ⌋*(RS_MANI_00061)*

Class	DiagnosticEventInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of diagnostic events on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.67: DiagnosticEventInterface

3.11.5 Interface to diagnostic Condition

[TPS_MANI_01249]{DRAFT} **Semantics of** `DiagnosticConditionInterface` ⌈ AUTOSAR supports different diagnostic conditions, i.e. enable condition, storage condition, and clear condition. This aspect is represented in the definition of the `DiagnosticConditionInterface` for the *AUTOSAR adaptive platform*. ⌋*(RS_MANI_00061)*

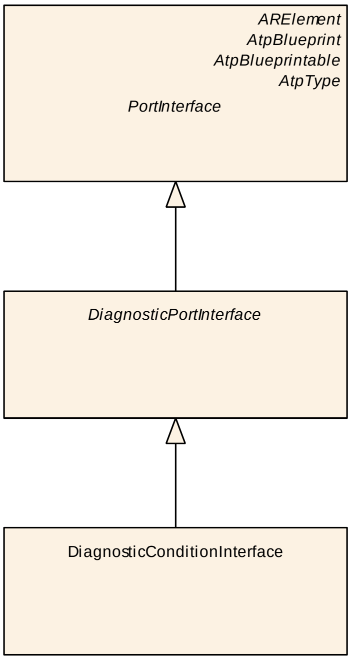


Figure 3.44: Modeling of `DiagnosticConditionInterface`

Class	**DiagnosticConditionInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for diagnostic conditions on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 3.68: DiagnosticConditionInterface

The `DiagnosticConditionInterface` does not require any further details in its formalization.

3.11.6 Indicator Interface

[TPS_MANI_01250]{DRAFT} **Semantics of `DiagnosticIndicatorInterface`**
⌈ The usage of the `DiagnosticIndicatorInterface` is foreseen for software that implements a diagnostic indicator (i.e. a warning light on the dashboard). ⌋ *(RS_MANI_00061)*

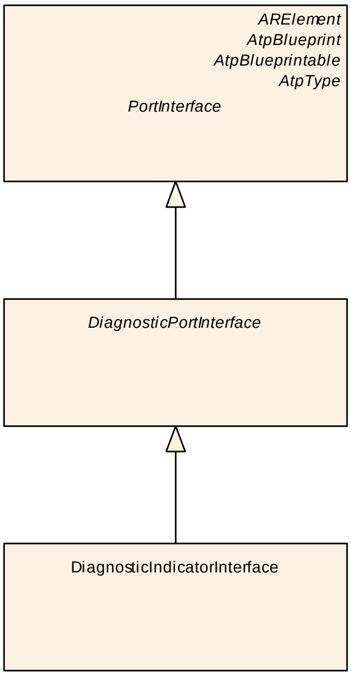


Figure 3.45: Modeling of DiagnosticIndicatorInterface

Class	**DiagnosticIndicatorInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement indicator functionality on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.69: DiagnosticIndicatorInterface

The DiagnosticIndicatorInterface does not require any further details in its formalization.

3.11.7 Security Level Interface

[TPS_MANI_01251]{DRAFT} **Semantics of DiagnosticSecurityLevelInterface** ⌈ The usage of the DiagnosticSecurityLevelInterface is foreseen for software that implements the checks for the clearance of a given security level. ⌋ *(RS_MANI_00061)*

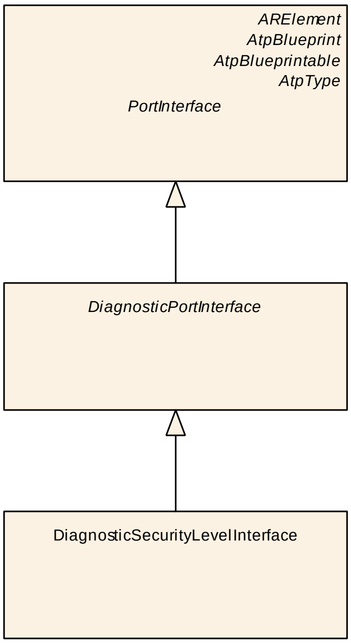


Figure 3.46: Modeling of DiagnosticSecurityLevelInterface

Class	DiagnosticSecurityLevelInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a security-level-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.70: DiagnosticSecurityLevelInterface

The DiagnosticSecurityLevelInterface does not require any further details in its formalization.

3.11.8 Service Validation Interface

[TPS_MANI_01252]{DRAFT} **Semantics of DiagnosticServiceValidationInterface** ⌈ The usage of the DiagnosticServiceValidationInterface is foreseen for software that implements the checks for clearance on manufacturer or supplier level. ⌋*(RS_MANI_00061)*

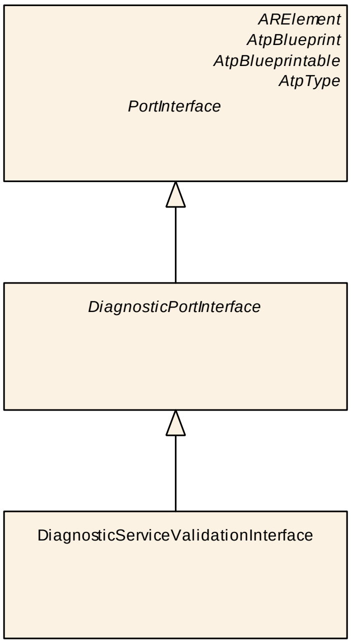


Figure 3.47: Modeling of DiagnosticServiceValidationInterface

Class	**DiagnosticServiceValidationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for service validation on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.71: DiagnosticServiceValidationInterface

The DiagnosticServiceValidationInterface does not require any further details in its formalization.

3.11.9 Operation Cycle Interface

[TPS_MANI_01253]{DRAFT} **Semantics of DiagnosticOperationCycleInterface** ⌈ The usage of the DiagnosticOperationCycleInterface is foreseen for software that implements the manages the operation cycles. ⌋*(RS_MANI_00061)*

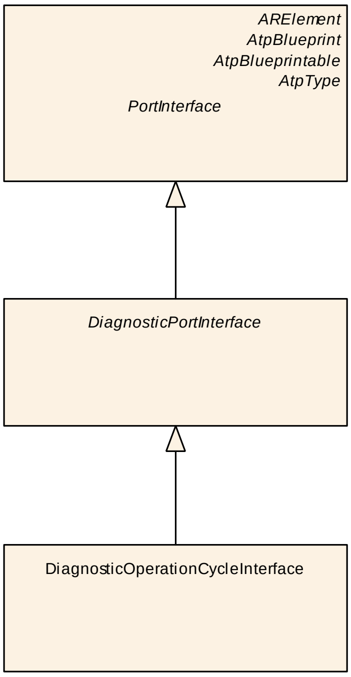


Figure 3.48: Modeling of DiagnosticOperationCycleInterface

Class	**DiagnosticOperationCycleInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for operation cycles on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 3.72: DiagnosticOperationCycleInterface

The DiagnosticOperationCycleInterface does not require any further details in its formalization.

3.11.10 Generic UDS Interface

[TPS_MANI_01254]{DRAFT} **Semantics of DiagnosticGenericUdsInterface** ⌈ The AUTOSAR diagnostic communication API also foresees the existence of one DiagnosticPortInterface that support the implementation of a completely generic handler of a UDS service. ⌋*(RS_MANI_00061)*

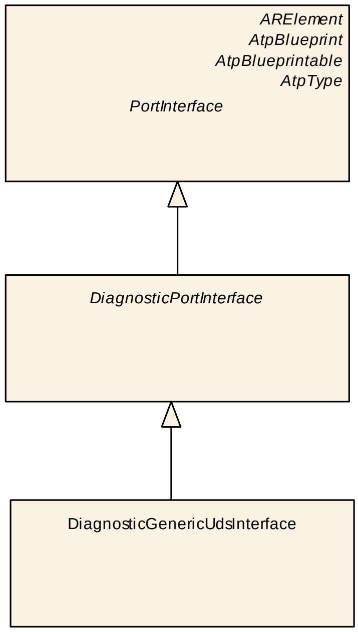


Figure 3.49: Modeling of DiagnosticGenericUdsInterface

Class	**DiagnosticGenericUdsInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic UDS PortInterface for diagnostics on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 3.73: DiagnosticGenericUdsInterface

The DiagnosticGenericUdsInterface does not require any further details in its formalization.

3.11.11 DoIP Interfaces

[TPS_MANI_01255]{DRAFT} **Semantics of DiagnosticGenericUdsInterface**
⌈ The AUTOSAR diagnostic communication API also foresees the existence of DiagnosticPortInterfaces to implement functionalities in the context of DoIP operation.

Specifically, the following concrete sub-classes of DiagnosticPortInterface are defined to support the implementation of functionalities in the context of DoIP:

- DiagnosticDoIPGroupIdentificationInterface
- DiagnosticDoIPPowerModeInterface

⌋(*RS_MANI_00061*)

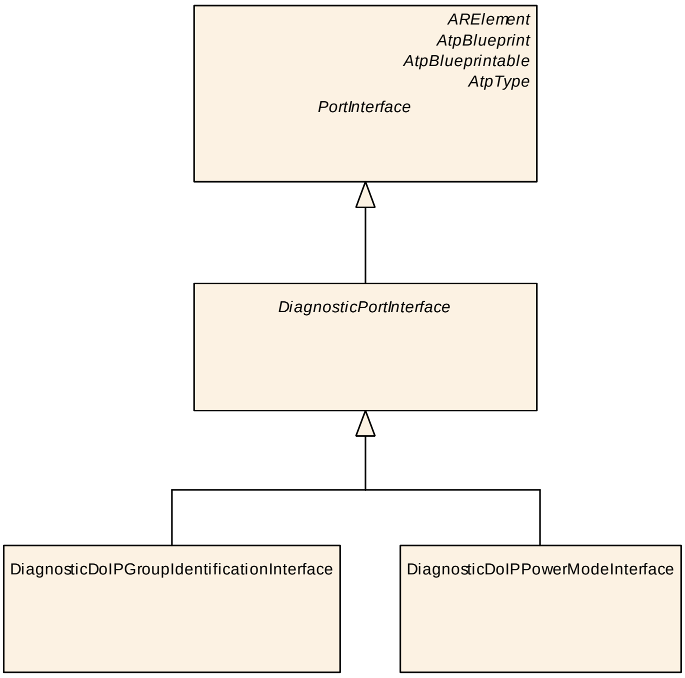


Figure 3.50: Modeling of `DiagnosticDoIPGroupIdentificationInterface` and `DiagnosticDoIPPowerModeInterface`

Class	**DiagnosticDoIPGroupIdentificationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Group Identification on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.74: DiagnosticDoIPGroupIdentificationInterface

Class	**DiagnosticDoIPPowerModeInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Power Mode on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*

▽

△

Class	DiagnosticDoIPPowerModeInterface			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 3.75: DiagnosticDoIPPowerModeInterface

The DiagnosticDoIPGroupIdentificationInterface and Diagnostic-DoIPPowerModeInterface do not require any further details in its formalization.

3.11.12 Diagnostic Interfaces for Upload and Download

[TPS_MANI_01265]{DRAFT} **Semantics of DiagnosticDownloadInterface and DiagnosticDownloadInterface** ⌈ The AUTOSAR diagnostic communication API also foresees the existence of DiagnosticPortInterfaces to implement upload and download via diagnostic channels.

Specifically, the following concrete sub-classes of DiagnosticPortInterface are defined to support the implementation of upload and download:

- DiagnosticUploadInterface
- DiagnosticDownloadInterface

⌋*(RS_MANI_00061)*

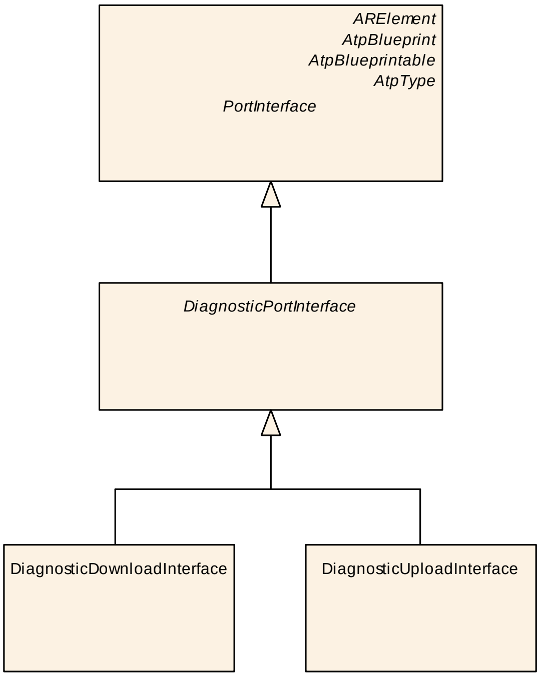


Figure 3.51: Modeling of DiagnosticUploadInterface and DiagnosticDownloadInterface

Class	**DiagnosticUploadInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for uploading data using diagnostic channels on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 3.76: DiagnosticUploadInterface

Class	**DiagnosticDownloadInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for downloading data using diagnostic channels on the adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 3.77: DiagnosticDownloadInterface

The `DiagnosticUploadInterface` and `DiagnosticDownloadInterface` do not require any further details in its formalization.

3.12 Interaction Endpoint for Application

The interaction of software-components with the outside world can take several forms, e.g. service-oriented communication or the interaction with a persistent data storage.

A formal representation of the interaction needs to be described as an anchor point for adding various additional configuration attributes that make sense in this context but would not make sense in the context of a `PortInterface`.

There is a model element that already has a long-standing tradition in the AUTOSAR meta-model for exactly the described purpose: the `PortPrototype`.

The following sub-chapters discuss the interaction by means of `PortPrototype`s with software “outside” a given software-component with the focus on different kinds of interaction that require different ways to further contribute model elements for configuration.

3.12.1 Service-oriented Communication

The service-oriented communication by means of PortPrototypes does **not** support the concept of a communication endpoint that is both required and provided **at the same time**. This motivates the existence of [constr_1473].

[constr_1473]{DRAFT} **No support for PRPortPrototype** ⌈ A ServiceInterface shall not be referenced by a PRPortPrototype in the role providedRequiredInterface. ⌋*()*

[TPS_MANI_01039]{DRAFT} **Representation of provided service** ⌈ A **provided service** shall be modeled by means of an PPortPrototype that is typed by a ServiceInterface. ⌋*(RS_MANI_00002)*

[TPS_MANI_01040]{DRAFT} **Representation of required service** ⌈ A **required service** shall be modeled by means of an RPortPrototype that is typed by a ServiceInterface. ⌋*(RS_MANI_00002)*

For more background regarding the rationale of [constr_1473], please refer to [1].

Please note that the utilization of service discovery on the *AUTOSAR adaptive platform* means that opposite communication ends **are by design not known upfront**.

As a consequence, it is in general not possible to use AssemblySwConnectors to model a pre-defined relation between two communication endpoints modeled as PortPrototypes.

Independent of the issue described above, it is still necessary to provide means for configuration of a given PortPrototype on different levels:

- The PortPrototype itself (i.e. as a whole) may need to be customized, independently of the kind or number of elements aggregated by the corresponding ServiceInterface. This aspect is discussed in section 3.12.4.
- The usage of elements of the corresponding ServiceInterface may need to be configured for a given PortPrototype. This aspect is discussed in section 3.12.5.

3.12.2 Interaction with Persistent Key-Value Storage

The usage of PortPrototypes for the purpose of interacting with *persistent key-value storage* is less restricted than in the case of service-oriented communication. In other words, it is perfectly valid to use a PRPortPrototype where applicable.

[TPS_MANI_01073]{DRAFT} **Semantics of PortPrototype typed by PersistencyKeyValueDatabaseInterface** ⌈ The usage of a specific sub-class of PortPrototype typed by PersistencyKeyValueDatabaseInterface indicates the intended semantics of interaction:

- The usage of a RPortPrototype indicates that the persistent data can only be **read from** the persistent storage.
- The usage of a PPortPrototype indicates that the persistent data can only be **written to** the persistent storage.
- The usage of a PRPortPrototype indicates that the persistent data can be **read from** as well as **written to** the persistent storage.

⌋*(RS_MANI_00027)*

3.12.3 Interaction with Persistent File-Based Storage

Interaction with **persistent file-based storage** can involve the ability to read from and write to a file by the same application. Therefore, the existence of a PRPortPrototype typed by a PersistencyFileProxyInterface shall be supported.

[TPS_MANI_01081]{DRAFT} **Semantics of PortPrototype typed by PersistencyFileProxyInterface** ⌈ The usage of a specific sub-class of PortPrototype typed by PersistencyFileProxyInterface indicates the intended semantics of interaction:

- The usage of a RPortPrototype indicates that the corresponding file(s) can be **opened for read access**.
- The usage of a PPortPrototype indicates that the corresponding file(s) can be **opened or created for write access**. Also, there is the ability to **delete** a file.
- The usage of a PRPortPrototype indicates that the corresponding file(s) can be **opened or created for read and write access**. Also, there is the ability to **delete** a file.

⌋*(RS_MANI_00027)*

3.12.4 Port Prototype Props

As mentioned before, in some cases a qualification of the semantics of PortPrototypes is necessary. For this purpose, AUTOSAR typically defines a *props* class of some kind. The same approach applies in this situation as well.

In particular, PortPrototype aggregates the abstract meta-class PortPrototypeProps, that in turn starts an inheritance tree of derived meta-classes that have the ability to qualify sub-classes of PortPrototype accordingly.

One example for this approach is the definition of the meta-class RPortPrototypeProps, sketched in Figure 3.52.

[constr_3359]{DRAFT} **RPortPrototypeProps are related only to RPortPrototypes** ⌈ The RPortPrototypeProps shall be aggregated only by a RPortPrototype in the role portPrototypeProps. ⌋*()*

[TPS_MANI_01052]{DRAFT} **Semantics of RPortPrototypeProps.portInstantiationBehavior** ⌈ The attribute RPortPrototypeProps.portInstantiationBehavior adds the ability to define whether a given RPortPrototype can have a "multiple-instantiation semantics".

This means that the RPortPrototype exists only as a single model-element but can have a collection-semantics in the implementation of the software-component. ⌋ *(RS_MANI_00002)*

[TPS_MANI_01057]{DRAFT} **Semantics of RPortPrototypeProps.searchBehavior** ⌈ The value of the attribute RPortPrototypeProps.searchBehavior clarifies whether the search for a corresponding offer shall be done as a search for "any" or else as a search for a specific ID.

Typically, a search for "any" results in a collection of offers while the search for a given id results in just a single offer. ⌋*(RS_MANI_00002)*

Please note that a search for "any" does not necessarily mean that [TPS_MANI_01052] applies, i.e. that the RPortPrototype is supposed to assume array semantics.

Even if a search for "any" is executed it may still be intended to select just a **single offer** from the result of the search. Therefore, the simultaneous existence of RPortPrototypeProps.searchBehavior and RPortPrototypeProps.portInstantiationBehavior is warranted.

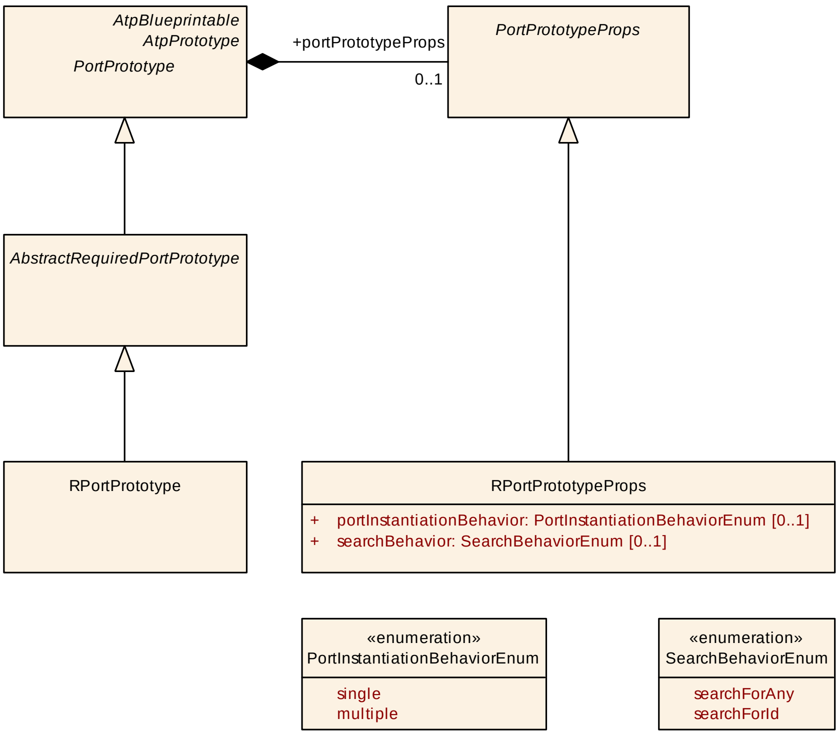


Figure 3.52: Modeling of the RPortPrototypeProps for RPortPrototype

Class	***PortPrototypeProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to define a further qualification of semantics of sub-classes of Port Prototype. **Tags:** atp.Status=draft			
Base	*ARObject*			
Subclasses	RPortPrototypeProps			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.78: PortPrototypeProps

Class	**RPortPrototypeProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	PortPrototypeProps for a RPort. **Tags:** atp.Status=draft			
Base	*ARObject*, *PortPrototypeProps*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
portInstantiation Behavior	PortInstantiation BehaviorEnum	0..1	attr	This attribute specifies how many proxy instances may be created at this RPort.
searchBehavior	SearchBehaviorEnum	0..1	attr	This attribute is used to specify the search behavior.

Table 3.79: RPortPrototypeProps

Enumeration	**PortInstantiationBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure
Note	This enumeration describes different option for the instantiation behavior of a PortPrototype. **Tags:** atp.Status=draft
Literal	***Description***
multiple	Multiple proxy instances may be created at this port. **Tags:** atp.EnumerationValue=1
single	A single proxy instance is created at this port **Tags:** atp.EnumerationValue=0

Table 3.80: PortInstantiationBehaviorEnum

Enumeration	**SearchBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign
Note	This meta-class allows for the definition of a dedicated search behavior from the application's point of view. **Tags:** atp.Status=draft
Literal	***Description***
searchForAny	This value represents the intention to search for "any" **Tags:** atp.EnumerationValue=0

▽

△

Enumeration	**SearchBehaviorEnum**
searchForId	This value represents the intention to search for a dedicated Id. **Tags:** atp.EnumerationValue=1

Table 3.81: SearchBehaviorEnum

3.12.5 Port Prototype ComSpec

[TPS_MANI_01053]{DRAFT} **Usage of `ComSpec`s on the *AUTOSAR adaptive platform*** ⌈ The aspect of further qualification of elements of the ServiceInterface used to type given PortPrototype is implemented by means of ComSpecs, i.e. specific sub-classes of the abstract meta-classes RPortComSpec and PPortComSpec.

However, the support for ComSpecs on the *AUTOSAR adaptive platform* only covers a **limited selection** of attributes of a specific ComSpec. ⌋*(RS_MANI_00002)*

The details about supported attributes of either a RPortComSpec or PPortComSpec are described in this chapter.

3.12.5.1 Port Prototypes typed by Service Interfaces

3.12.5.1.1 Receiver ComSpec

It is necessary to provide means to configure the queue length of the reception of an event on a case-by-case basis. In other words, even two "adjacent" events within the same RPortPrototype may need a different handling of the queue length.

[TPS_MANI_01054]{DRAFT} **Definition of the queue length of an `event` or `field` notifier** ⌈ The definition of the queue length of an event or field notifier shall be modeled by means of the attribute QueuedReceiverComSpec.queueLength. ⌋ *(RS_MANI_00026)*

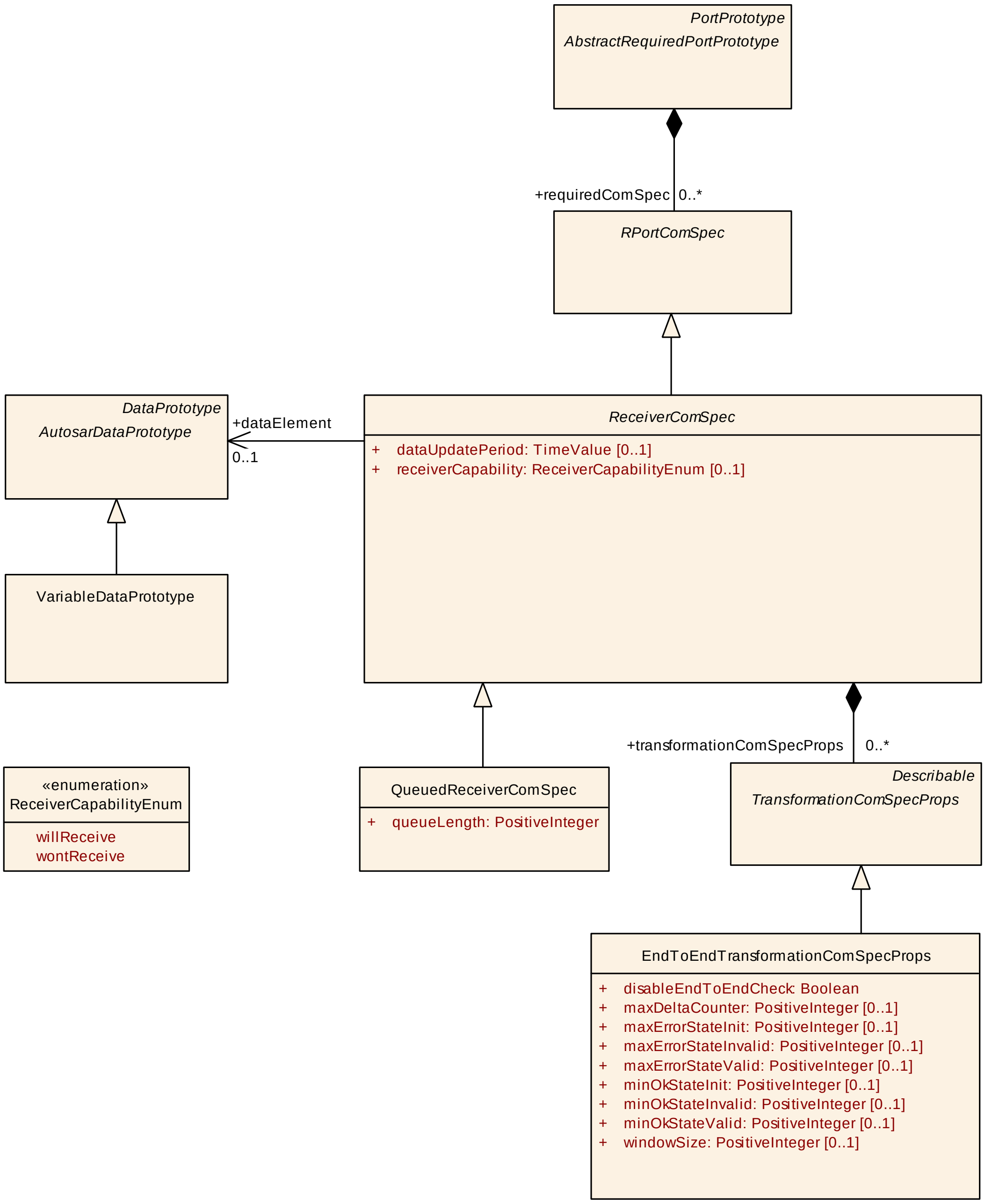


Figure 3.53: Modeling of the `ReceiverComSpec` on the *AUTOSAR adaptive platform*

Class	***ReceiverComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Receiver-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers.			
Base	*ARObject*, *RPortComSpec*			
Subclasses	NonqueuedReceiverComSpec, QueuedReceiverComSpec			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	***ReceiverComSpec*** (abstract)			
dataElement	AutosarDataPrototype	0..1	ref	Data element these attributes belong to.
dataUpdate Period	TimeValue	0..1	attr	This attribute defines the period in which the application shall check for updated data. This attribute is used for the configuration of the E2E protection. **Tags:** atp.Status=draft
receiver Capability	ReceiverCapability Enum	0..1	attr	This attribute represents the expressed capability of the receiver. The receiver may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific receiver. The conceptual background of this claim may be driven by security, safety, etc. **Tags:** atp.Status=draft
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.82: ReceiverComSpec

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

Table 3.83: QueuedReceiverComSpec

[TPS_MANI_01106]{DRAFT} **Specification of capabilities for the receiver of `events` or `field` notifiers** ⌈The attribute `ReceiverComSpec.receiverCapability` can be used to specify whether the software actually intends to access the referenced `event`s or `field` notifier or whether it explicitly states that it is not interested in the value. ⌋*(RS_MANI_00034)*

Enumeration	**ReceiverCapabilityEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the ability to specify how a given ServiceInterface is used from the perspective of a given event receiver. **Tags:** atp.Status=draft
Literal	***Description***
willReceive	The receiver will receive the event or field notifier. **Tags:** atp.EnumerationValue=0
wontReceive	The receiver won't receive the event or field notifier. **Tags:** atp.EnumerationValue=1

Table 3.84: ReceiverCapabilityEnum

[TPS_MANI_03132]{DRAFT} **Semantics of E2E attributes in `ReceiverComSpec`** ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification of `RPortPrototype`-specific configuration options related to end-to-end protection of events or field notifiers. ⌋*(RS_MANI_00028)*

Class	**EndToEndTransformationComSpecProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	The class EndToEndTransformationIComSpecProps specifies port specific configuration properties for EndToEnd transformer attributes.			
Base	*ARObject*, *Describable*, *TransformationComSpecProps*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
disableEndToEndCheck	Boolean	1	attr	Disables/Enables the E2E check. The E2Eheader is removed from the payload independent from the setting of this attribute.
maxDeltaCounter	PositiveInteger	0..1	attr	Maximum allowed difference between two counter values of two consecutively received valid messages. For example, if the receiver gets data with counter 1 and MaxDeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
maxErrorStateInit	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_INIT. The minimum value is 0.
maxErrorStateInvalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_INVALID. The minimum value is 0.
maxErrorStateValid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_VALID. The minimum value is 0.
minOkStateInit	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INIT. The minimum value is 1.
minOkStateInvalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID. The minimum value is 1.
minOkStateValid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID. The minimum value is 1.
windowSize	PositiveInteger	0..1	attr	Size of the monitoring window for the E2E state machine. The meaning is the number of correct cycles (E2E_P_OK) that are required in E2E_SM_INITCOM before the transition to E2E_SM_VALID. The minimum allowed value is 1.

Table 3.85: EndToEndTransformationComSpecProps

3.12.5.1.2 Sender ComSpec

The `SenderComSpec` is modeled in the same way as described in the Software Component Template [1]. It has some specific additions, e.g. the introduction of the attribute `dataUpdatePeriod` that defines the frequency with which the data is updated by the application.

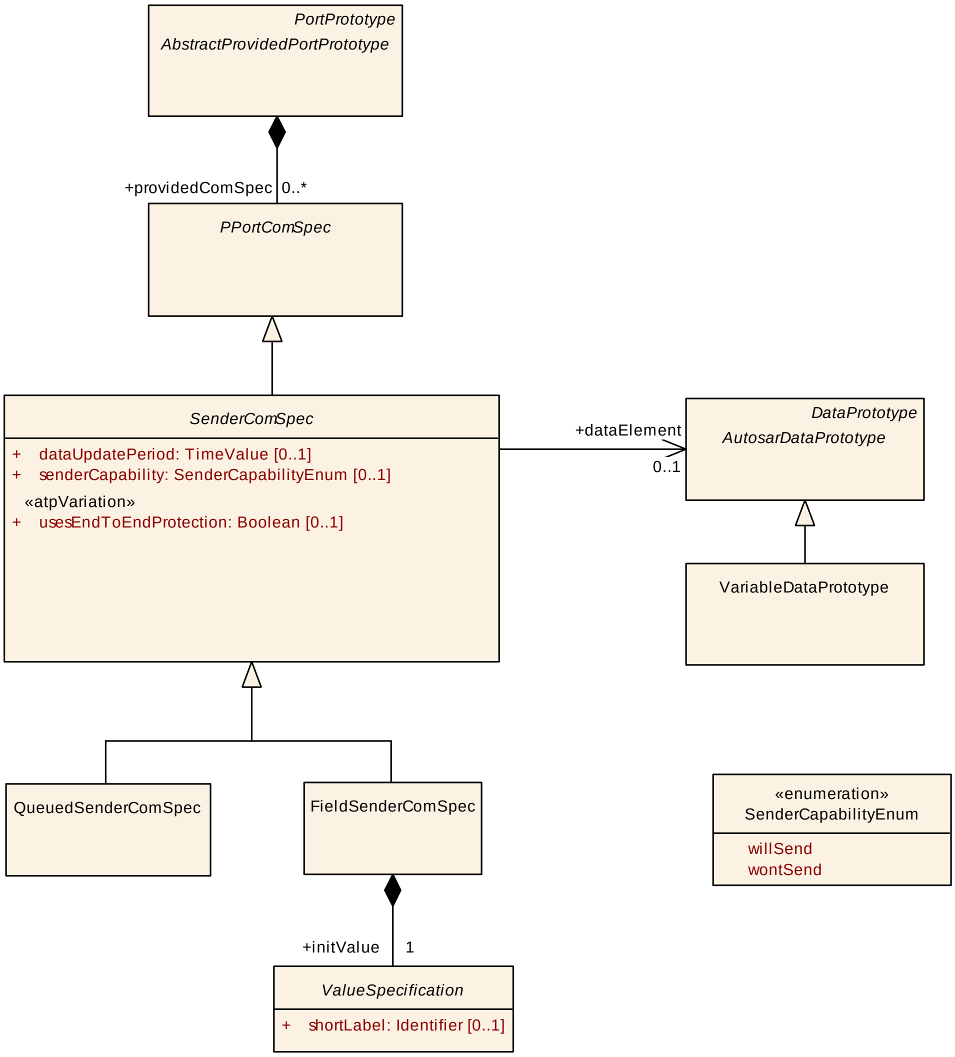


Figure 3.54: Modeling of the SenderComSpec on the *AUTOSAR adaptive platform*

Class	***SenderComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for a sender port (PPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers.			
Base	*ARObject*, *PPortComSpec*			
Subclasses	FieldSenderComSpec, NonqueuedSenderComSpec, QueuedSenderComSpec			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataElement	AutosarDataPrototype	0..1	ref	Data element these quality of service attributes apply to.
dataUpdate Period	TimeValue	0..1	attr	This attribute describes the period in which the applications are assumed to transmit E2E-protected messages. The middleware does not use this attribute at all. **Tags:** atp.Status=draft

△

Class	***SenderComSpec*** (abstract)			
sender Capability	SenderCapabilityEnum	0..1	attr	This attribute represents the expressed capability of the sender. The sender may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific sender. The conceptual background of this claim may be driven by security, safety, etc. **Tags:** atp.Status=draft
usesEndToEnd Protection	Boolean	0..1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table 3.86: SenderComSpec

[TPS_MANI_03210]{DRAFT} **Specification of `event` specific communication attributes** ⌈ The meta-class `QueuedSenderComSpec` can be used to specify communication attributes that are relevant for an `event` on the sender side. ⌋*(RS_MANI_00002)*

Class	**QueuedSenderComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to distribution of events (PPortPrototype, SenderReceiverInterface and dataElement carries an "event").			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.87: QueuedSenderComSpec

[TPS_MANI_03211]{DRAFT} **Specification of `field` specific communication attributes** ⌈ The meta-class `FieldSenderComSpec` can be used to specify communication attributes that are relevant for a `field` on the sender side. ⌋*(RS_MANI_00002)*

Class	**FieldSenderComSpec**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	Port specific communication attributes for a Field that is defined in a ServiceInterface. **Tags:** atp.Status=draft			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
initValue	ValueSpecification	1	aggr	Initial value for a Field that is set before the Service Interface is offered. **Tags:** atp.Status=draft

Table 3.88: FieldSenderComSpec

[TPS_MANI_03212]{DRAFT} **Specification of initial value for a `field`** ⌈ The attribute `FieldSenderComSpec.initValue` can be used to specify an initial Value for a `field`. ⌋*(RS_MANI_00002)*

A `field` has a valid value at any time as described in subsection 3.4.3. ara::com ensures that a service implementation providing a field has a field value before the field becomes visible to potential consumers. This is explained in more detail in [13] where it is defined that the initial field value must be set at least once via Update() by the application code before OfferService() gets called. Custom-code (e.g. component model above ara::com) may use the defined initValue to call Field.Update(initValue).

[TPS_MANI_01107]{DRAFT} **Specification of capabilities for the sender of events or field notifiers** ⌈ The attribute SenderComSpec.senderCapability can be used to specify whether the software actually intends to send the referenced events or field notifier. ⌋*(RS_MANI_00034)*

Enumeration	**SenderCapabilityEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the ability to specify how a given ServiceInterface is used from the perspective of a given event sender. **Tags:** atp.Status=draft
Literal	*Description*
willSend	The sender will send the event or field notifier. **Tags:** atp.EnumerationValue=0
wontSend	The sender won't send the event or field notifier. **Tags:** atp.EnumerationValue=1

Table 3.89: SenderCapabilityEnum

3.12.5.1.3 Client ComSpec

The ClientComSpec undergoes extensions for the *AUTOSAR adaptive platform*, namely the ability to refer to the getter and setter method of a field and the definition of capabilities.

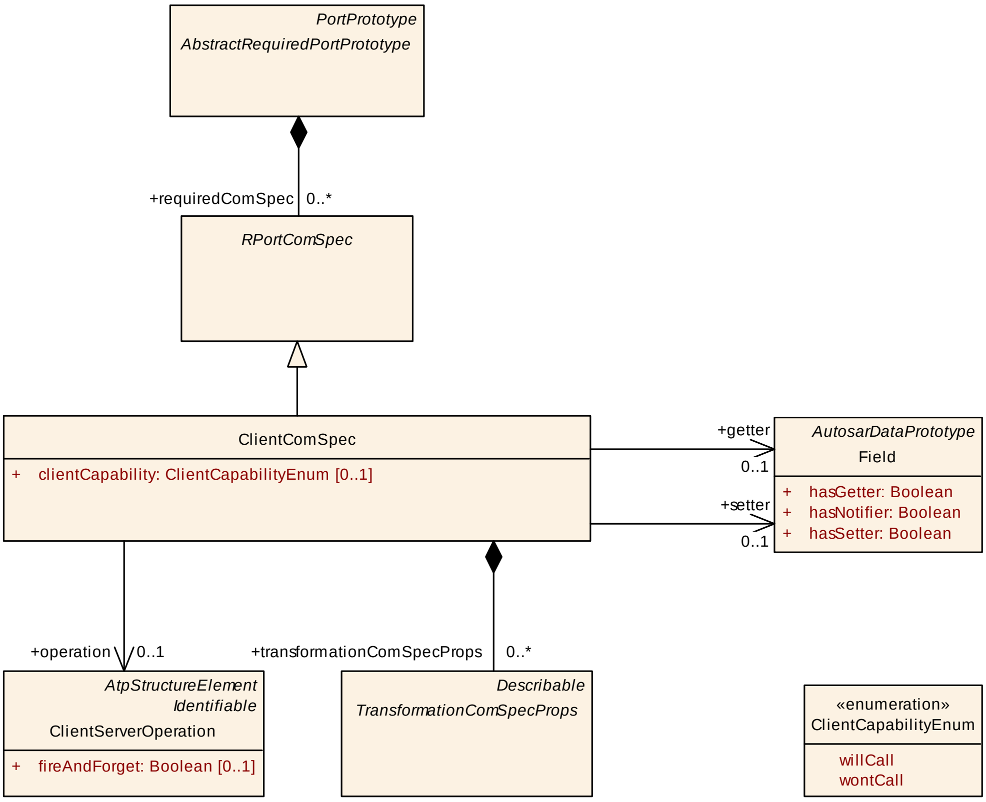


Figure 3.55: Modeling of the `ClientComSpec` on the *AUTOSAR adaptive platform*

Class	**ClientComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Client-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for methods and field getters and setters.			
Base	*ARObject*, *RPortComSpec*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
clientCapability	ClientCapabilityEnum	0..1	attr	This attribute represents the expressed capability of the client. The client may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific client. The conceptual background of this claim may be driven by security, safety, etc. **Tags:** atp.Status=draft
getter	Field	0..1	ref	The existence of this reference indicates that the Client ComSpec refers to the getter of a Field. **Tags:** atp.Status=draft
operation	ClientServerOperation	0..1	ref	This represents the corresponding ClientServerOperation.
setter	Field	0..1	ref	The existence of this reference indicates that the Client ComSpec refers to the setter of a Field. **Tags:** atp.Status=draft
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.90: ClientComSpec

[TPS_MANI_01108]{DRAFT} **Specification of capabilities for the caller of a methods or field setter/getter** ⌈ The attribute ClientComSpec.clientCapability can be used to specify whether the software actually intends to call the referenced methods or getter/setter of a referenced field. ⌋*(RS_MANI_00034)*

Enumeration	**ClientCapabilityEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the ability to specify how a given ServiceInterface is used from the perspective of a given client. **Tags:** atp.Status=draft
Literal	*Description*
willCall	The client will call this method. **Tags:** atp.EnumerationValue=0
wontCall	The client won't call this method. **Tags:** atp.EnumerationValue=1

Table 3.91: ClientCapabilityEnum

Please note that the existence of the ServerComSpec has not explicitly been mentioned in this chapter because there is no extension or additional attribute that needs documentation for the *AUTOSAR adaptive platform*.

3.12.5.2 Port Prototypes typed by Persistency Data Interfaces

[TPS_MANI_01069]{DRAFT} **Further qualification of properties of PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces** ⌈ For PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces it is possible to define further qualifying attributes for the provider side.

For this purpose meta-class PersistencyDataProvidedComSpec is provided. ⌋ *(RS_MANI_00027)*

Class	**PersistencyDataProvidedComSpec**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	This meta-class represents the ability to define port-specific attributes for supporting use cases of data persistency on the provided side. **Tags:** atp.Status=draft			
Base	*ARObject*, *PPortComSpec*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
dataElement	PersistencyDataElement	1	ref	This refrence represents the PersistencyDataElement for which the PersistencyDataProvidedComSpec applies. **Tags:** atp.Status=draft

▽

△

Class	**PersistencyDataProvidedComSpec**			
initValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of an initial value for the PersistencyDataElement referenced by the enclosing PersistencyDataProvidedComSpec **Tags:** atp.Status=draft

Table 3.92: PersistencyDataProvidedComSpec

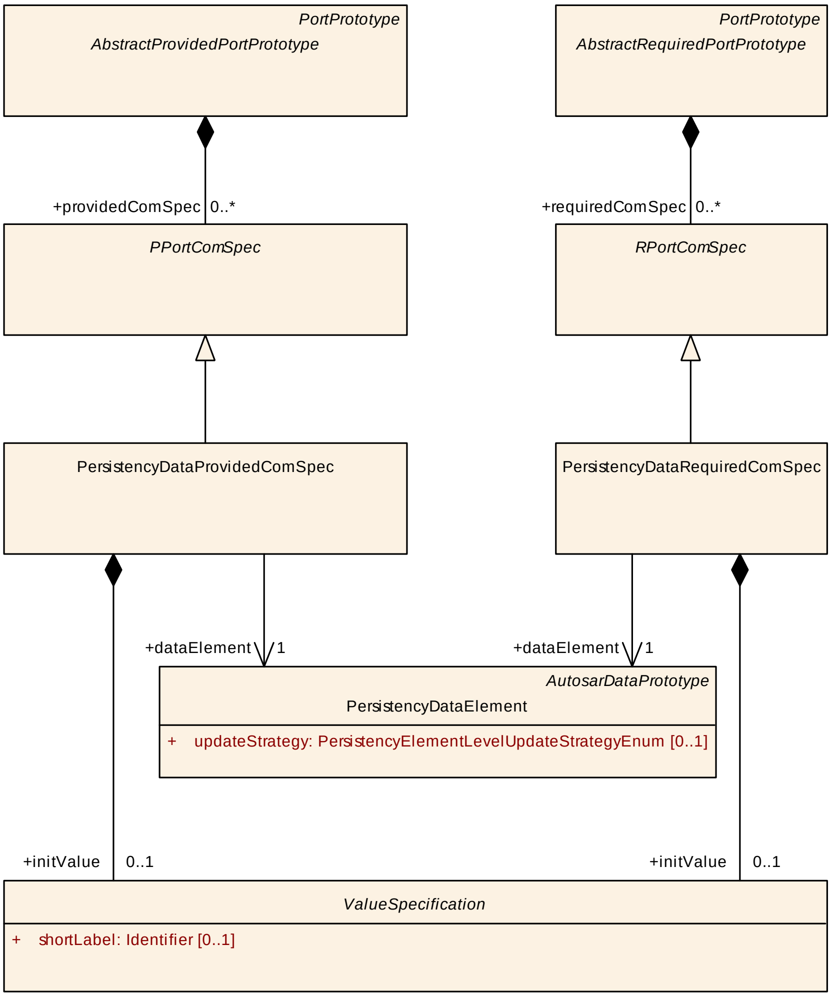


Figure 3.56: Modeling of `ComSpec` for persistency

Note that the specification of encryption as described in [TPS_MANI_01073] does not impose a binding contract. An integrator may reasonably have various reasons to overrule the configuration in the `PersistencyDataProvidedComSpec`.

It would simply not make any sense to statically model the encryption algorithms by means of an enumeration in the AUTOSAR meta-model and consequently require an update of this very enumeration in the AUTOSAR meta-model and XML schema in order to be able to use that hipster encryption algorithm that happens to fulfill ambitious needs in terms of encryption for a specific purpose.

[TPS_MANI_01160]{DRAFT} **Definition of initial value for Persistency-DataElement** ⌈ The definition of an initial value for a PersistencyDataElement can be done on the level of a PortPrototype by means of either PersistencyDataProvidedComSpec.initValue or PersistencyDataRequiredComSpec.initValue ⌋*(RS_MANI_00027)*

Class	PersistencyDataRequiredComSpec			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	This meta-class represents the ability to define port-specific attributes for supporting use cases of data persistency on the required side. **Tags:** atp.Status=draft			
Base	*ARObject*, *RPortComSpec*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataElement	PersistencyDataElement	1	ref	This refrence represents the PersistencyDataElement for which the PersistencyDataRequiredComSpec applies. **Tags:** atp.Status=draft
initValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of an initial value for the PersistencyDataElement referenced by the enclosing PersistencyDataRequiredComSpec **Tags:** atp.Status=draft

Table 3.93: PersistencyDataRequiredComSpec

3.13 Executable

[TPS_MANI_01010]{DRAFT} **Root element for a hierarchical software-component** ⌈ Executable aggregates meta-class RootSwComponentPrototype in the role rootSwComponentPrototype to provide a root element for an arbitrarily nested hierarchy of software-components represented by the reference RootSwComponentPrototype.applicationType. ⌋*(RS_MANI_00004)*

Please note that the aggregation of RootSwComponentPrototype by Executable is the basis for the applicability of an ≪instanceRef≫ reference into the hierarchy of software-components that represent the functionality of the Executable.

This modeling approach is similar to the modeling of a System on the *AUTOSAR classic platform*.

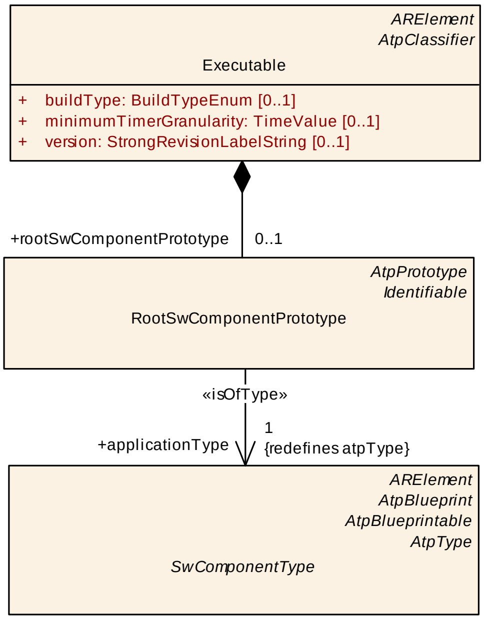


Figure 3.57: Modeling of the Executable

Class	**Executable**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents an executable program. **Tags:** atp.Status=draft atp.recommendedPackage=Executables			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
buildType	BuildTypeEnum	0..1	attr	This attribute describes the buildType of a module and/or platform implementation.
minimumTimer Granularity	TimeValue	0..1	attr	This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable. **Tags:** atp.Status=draft
rootSw Component Prototype	RootSwComponent Prototype	0..1	aggr	This represents the root SwCompositionPrototype of the Executable. This aggregation is required (in contrast to a direct reference of a SwComponentType) in order to support the definition of instanceRefs in Executable context. **Tags:** atp.Status=draft
version	StrongRevisionLabel String	0..1	attr	Version of the executable. **Tags:** atp.Status=draft

Table 3.94: Executable

Enumeration	BuildTypeEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation
Note	This enumeration defines the possible buildTypes a software module may be implemented. **Tags:** atp.Status=draft
Literal	***Description***
buildTypeDebug	Used for debugging. **Tags:** atp.EnumerationValue=1
buildTypeRelease	Used for releasing. **Tags:** atp.EnumerationValue=0

Table 3.95: BuildTypeEnum

[constr_1605]{DRAFT} **Standardized values of attribute Executable.category** ⌈ The following values for attribute Executable.category are standardized by AUTOSAR:

- PLATFORM_LEVEL: the Executable represents software on the platform level (i.e. conceptually located *on the level of* the middleware).
- APPLICATION_LEVEL: the Executable represents software on the application level (i.e. conceptually located *above* the middleware).

⌋*()*

[TPS_MANI_03056]{DRAFT} **Optionality of Executable.rootSwComponentPrototype** ⌈ The aggregation Executable.rootSwComponentPrototype has been made optional in order to support the implementation of *platform modules* that do not utilize any service oriented communication and don't require any further formalization. ⌋*(RS_MANI_00023)*

[constr_1492]{DRAFT} **SwComponentType referenced as Executable.rootSwComponentPrototype.applicationType** ⌈ Any SwComponentType referenced in the role Executable.rootSwComponentPrototype.applicationType, or used to type a SwComponentPrototype nested inside the SwComponentType referenced in the role Executable.rootSwComponentPrototype.applicationType shall **only** be either a CompositionSwComponentType or an AdaptiveApplicationSwComponentType. ⌋*()*

The example depicted in Figure 3.58 exemplifies the statement of [constr_1492]. The example shows a component hierarchy that consists of SwComponentPrototypes that are excursively typed by either a CompositionSwComponentType or an AdaptiveApplicationSwComponentType.

While the left part of Figure 3.58 resembles the modeling in the meta-model, the right part uses a simplified notation to give an idea how the nested definition of software-components could look like.

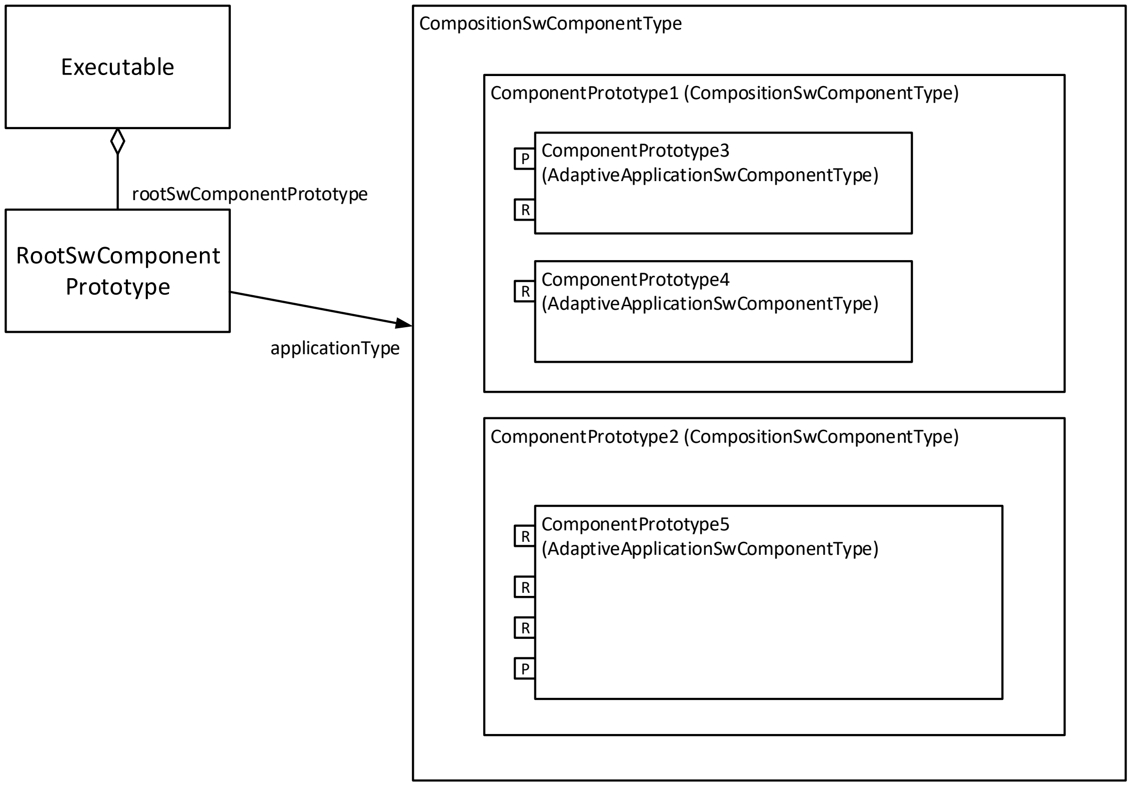


Figure 3.58: Example of the possible structure of an `Executable`

An obvious consequence of [constr_1492] is that no software-component that could be used on the *AUTOSAR classic platform* is allowed on the *AUTOSAR adaptive platform*, i.e. in the context of an `Executable.rootSwComponentPrototype.applicationType`.

Software-components on the *AUTOSAR adaptive platform* are mainly defined by their interaction with the outside world by means of `PortPrototype`s typed by `ServiceInterface`s. The definition of an internal behavior, with a minor exception, is not foreseen.

This lack of internal structure, in combination with decisions made regarding the scope of the generation of header files, leads to a situation where the implementation of a software component in source code is (in comparison to the situation on the *AUTOSAR classic platform*) way less subject to a strict separation.

In other words, there is no real motivation to implement software-components separately from each other. It would be possible, although not encouraged, to implement all software-components of a given executable program directly within the `Main()` function of the program.

Class	RootSwComponentPrototype			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within an Executable. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, etc.). **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
applicationType	SwComponentType	1	tref	This SwComponentType acts as the Type of the RootSw ComponentPrototype. **Stereotypes:** isOfType **Tags:** atp.Status=draft

Table 3.96: RootSwComponentPrototype

Class	***SwComponentType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AdaptiveApplicationSwComponentType, *AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
port	PortPrototype	*	aggr	The PortPrototypes through which this SwComponent Type can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComponentDocumentation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10

Table 3.97: SwComponentType

Class	CompositionSwComponentType
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition

▽

△

Class	**CompositionSwComponentType**			
Note	A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by Sw ComponentTypes) as well as SwConnectors for primarily connecting SwComponentPrototypes among each others and towards the surface of the CompositionSwComponentType. By this means hierarchical structures of software-components can be created. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortCom Spec. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the used ApplicationDataTypes in ServiceInterfaces. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dataTypeMapping

Table 3.98: CompositionSwComponentType

3.14 Optional Members in complex Data Structures

3.14.1 Background

The *AUTOSAR adaptive platform* supports the usage of a `TLV`[7] data encoding on the SOME/IP transport layer. `TLV` is typically used where at least a part of the transmitted data is only *optionally* existing and filled with meaningful values.

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

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

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

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

[7] This abbreviation stands for tag-length-value

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

AutosarDataType In this case the optionality that is primarily only needed for communication purposes would still be existing in all other usages of data types. AUTOSAR still sees use cases for implementing this option, especially in the context of the *AUTOSAR classic platform*.

Admittedly, the definition of different optionality configurations for the same data type may lead to the existence of a bunch of structurally identical data types that only vary in terms of optionality. The existence of variation points may help to mitigate this effect, though.

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

This would lead to an increased effort for the definition of C++ data types in the context of the same ServiceInterface. Additional constraints have been identified in the context of the *AUTOSAR classic platform* that finally render this option as not viable.

ComSpec In this case the definition of optionality would even be more specific in comparison to the definition of optionality on the level of ServiceInterfaces.

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

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

3.14.2 Definition of Optionality

As mentioned before, the concrete definition of optionality on the level of an AutosarDataType is done by the indication of individual elements of the composite AutosarDataType.

More specifically, the definition of optionality needs to be supported for subclasses of AutosarDataType, namely on the level of ApplicationDataType as well as on the level of CppImplementationDataType.

In other words, if ApplicationDataTypes with optional elements are used to define a ServiceInterface then it is still necessary to convey the optionality down to the level of data type definition that directly affects the language binding of the AUTOSAR model.

Figure 3.59 shows the modeling of optionality on the level of ApplicationDataType.

[TPS_MANI_01184]{DRAFT} **Definition of optional elements on the level of ApplicationDataType** ⌈ The modeling approach for the definition of optional elements on the level of ApplicationDataType is to set the attribute ApplicationRecordElement.isOptional to the value True.

If the attribute is not set or set to the value False then the respective ApplicationRecordElement **shall be considered mandatory.** ⌋*(RS_MANI_00030)*

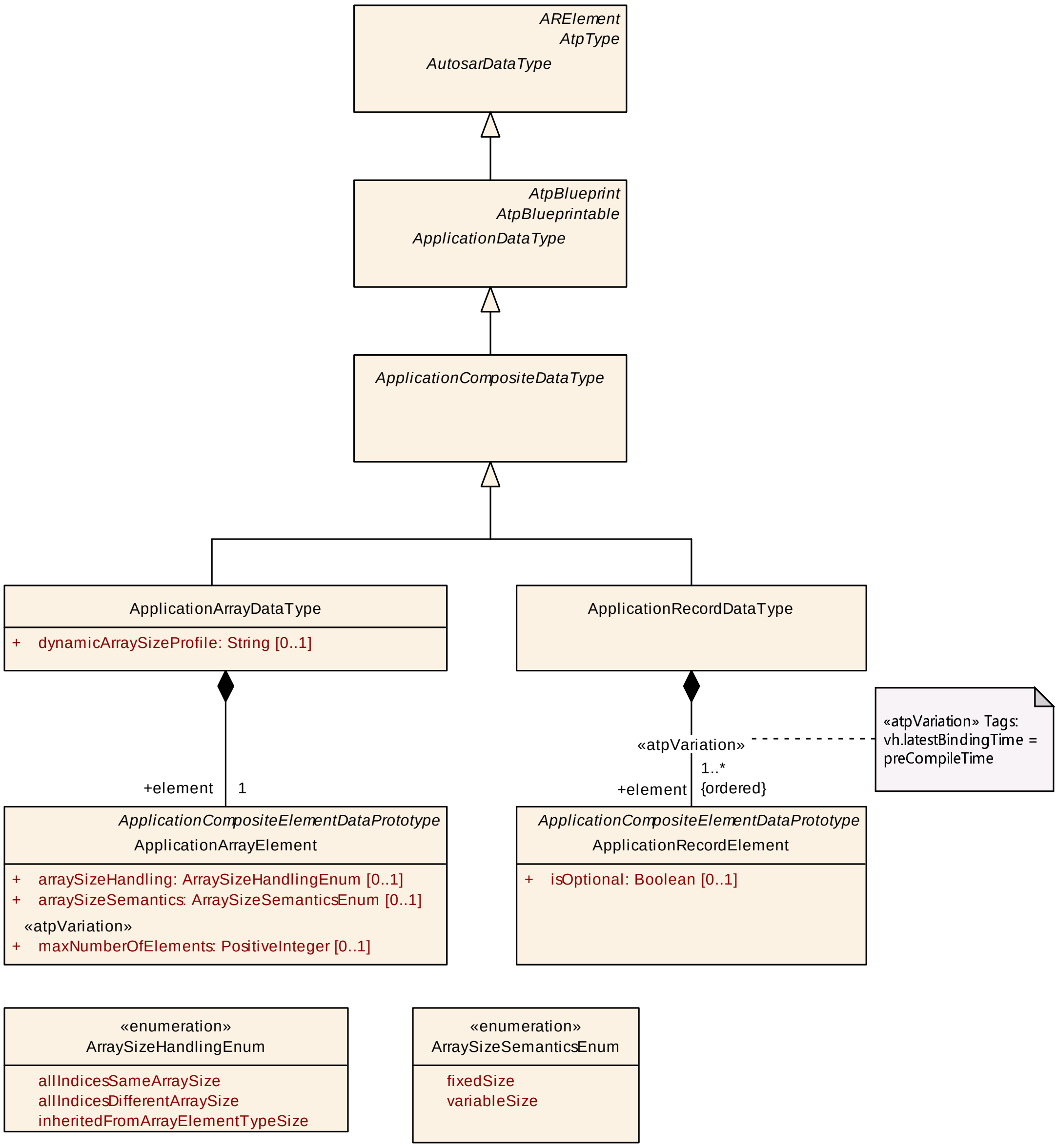


Figure 3.59: Modeling of optionality on the level of ApplicationDataType

Class	ApplicationRecordDataType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which can be decomposed into prototypes of other application data types. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
element (ordered)	ApplicationRecordElement	1..*	aggr	Specifies an element of a record. The aggregation of ApplicationRecordElement is subject to variability with the purpose to support the conditional existence of elements inside a ApplicationrecordDataType. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table 3.99: ApplicationRecordDataType

Class	ApplicationRecordElement			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Describes the properties of one particular element of an application record data type.			
Base	*ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ApplicationRecordElement as optional. This means the that, at runtime, the ApplicationRecordElement may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the ApplicationRecordElement as not valid at the sending end of a communication and determine its validity at the receiving end. **Tags:** atp.Status=draft

Table 3.100: ApplicationRecordElement

On top of that, it is still possible to use `CppImplementationDataType` directly for the definition of a `ServiceInterface`.

[TPS_MANI_01185]{DRAFT} **Definition of optional elements on the level of `CppImplementationDataType`** ⌈ The modeling approach for the definition of optional elements on the level of `CppImplementationDataType` is to set the attribute `CppImplementationDataTypeElement.isOptional` to the value `True`.

If the attribute is not set or set to the value `False` then the respective `CppImplementationDataTypeElement` **shall be considered mandatory**. ⌋*(RS_MANI_00030)*

3.15 Serialization Properties

In Adaptive AUTOSAR, the serialization code is generated out of the service description and is compiled and executed in the application context.

The meta-class TransformationPropsToServiceInterfaceElementMapping defines the serialization for a ServiceInterface element and provides the necessary serialization settings with the TransformationProps element.

The existence of a TransformationPropsToServiceInterfaceElementMapping demands the existence of serialization code that is linked with the application component object file to an application binary.

The serialization of SOME/IP is based on the ServiceInterface specification. If an AutosarDataPrototype that is used within a ServiceInterface is composite like a structure, union or array then SOME/IP supports the configuration of length fields that will be put in front of the serialized data.

AUTOSAR supports the configuration of such serialization settings on two different levels:

- Modeling on ServiceInterface element level that is valid for all available occurrences of a DataPrototype in the ServiceInterface element. This case is described in detail in chapter 3.15.1.
- Fine granular modeling on the level of DataPrototypes described in this chapter. This case is described in detail in chapter 3.15.2.

3.15.1 Default Values for Serialization Properties

[TPS_MANI_03101]{DRAFT} **SOME/IP serialization** ⌈ The ApSomeipTransformationProps meta-class that is referenced by the TransformationPropsToServiceInterfaceElementMapping in the role transformationProps provides the ability to define a SOME/IP serialization settings for ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[constr_3395]{DRAFT} **TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface** ⌈ All ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field shall be aggregated by the same ServiceInterface in the role event, method or field. ⌋*()*

[TPS_MANI_03103]{DRAFT} **Default size for all array and map length fields** ⌈ The attribute sizeOfArrayLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by

SOME/IP in front of all available variable size arrays (vectors), fixed size arrays and associative_maps defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

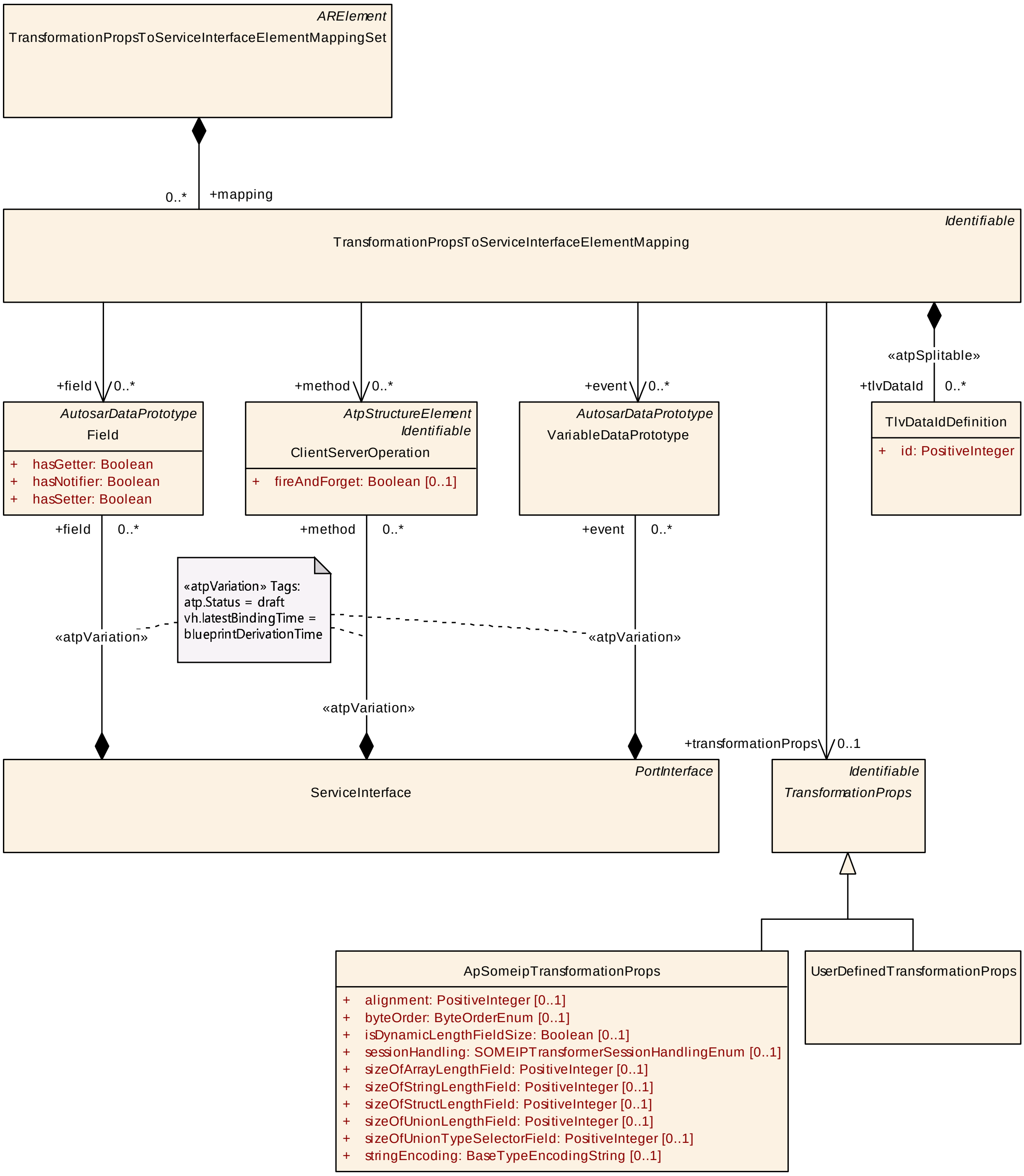


Figure 3.60: Association of serialization properties with a ServiceInterface

[TPS_MANI_03104]{DRAFT} **Default size for all structure length fields** ⌈ The attribute sizeOfStructLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available structures defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03117]{DRAFT} **Default size for all string length fields** ⌈ The attribute sizeOfStringLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available strings defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03105]{DRAFT} **Default size for all union length fields** ⌈ The attribute sizeOfUnionLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available unions defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03106]{DRAFT} **Default size for all union type selector fields** ⌈ The attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a type field generated by SOME/IP in front of all available unions defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03107]{DRAFT} **Default alignment for all dynamic DataPrototypes** ⌈ The attribute alignment of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of all variable data length data elements defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03108]{DRAFT} **Default Byte Order for all DataPrototypes** ⌈ The attribute byteOrder of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the Byte Order in the serialized data stream resulting from ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[constr_1614]{DRAFT} **Existence of attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.sessionHandling** ⌈

The attribute ApSomeipTransformationProps.sessionHandling shall only exist if the TransformationPropsToServiceInterfaceElementMapping that refers to the respective ApSomeipTransformationProps in the role transformationProps does **not** refer to a ClientServerOperation in the role method. ⌋*()*

[TPS_MANI_01210]{DRAFT} **Default encoding for all DataPrototypes typed by CppImplementationDataType of category STRING** ⌈ The attribute stringEncoding of a ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the string encoding in the serialized data stream resulting from ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋ *(RS_MANI_00008, RS_MANI_00025)*

[constr_1675]{DRAFT} **Existence of attribute ApSomeipTransformationProps.stringEncoding** ⌈ The attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.stringEncoding shall only exist for a event, method or field (referenced by the same TransformationPropsToServiceInterfaceElementMapping) that consists of or contains a DataPrototype typed by a CppImplementationDataType of category STRING. ⌋ *()*

Please note that more details about ApSomeipTransformationProps can be found in chapter 3.15.2.

[constr_1678]{DRAFT} **Allowed values for attribute ApSomeipTransformationProps.stringEncoding** ⌈ Imposed by technical restrictions in the definition of the SOME/IP message format [8], only two possible values of attribute ApSomeipTransformationProps.stringEncoding are allowed:

- UTF-8: UCS Transformation Format 8
- UTF-16: Character encoding for Unicode *code points* based on 16 bit *code units* [15]

⌋*()*

Class	**ApSomeipTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties			
Note	SOME/IP serialization properties. **Tags:** atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, TransformationProps*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
alignment	PositiveInteger	0..1	attr	Specifies the alignment of dynamic data in the serialized data stream. The alignment is specified in Bits.
byteOrder	ByteOrderEnum	0..1	attr	Specifies the byte order of data in the serialized data stream.

▽

△

Class	ApSomeipTransformationProps			
isDynamicLengthFieldSize	Boolean	0..1	attr	This attribute represents the ability to control the setting of the wire type for TLV encoding. If the attribute is set to True then wire type 5-7 shall be used. If the attribute does not exist or is set to False then wire type 4 shall be used.
sessionHandling	SOMEIPTransformerSessionHandlingEnum	0..1	attr	Defines whether the SOME/IP transformer shall use session handling for Sender/Receiver communication.
sizeOfArrayLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a variable size Array (Vector), fixed-size Array or an Associative_Map. It describes the size of the length field (in Bytes) that will be put in front of the Array or Associative_Map in the SOME/IP message.
sizeOfStringLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a String. It describes the size of the length field (in Bytes) that will be put in front of the String in the SOME/IP message.
sizeOfStructLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of an Struct. It describes the size of the length field (in Bytes) that will be put in front of the Struct in the SOME/IP message.
sizeOfUnionLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the length field (in Bytes) that will be put in front of the Union in the SOME/IP message.
sizeOfUnionTypeSelectorField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the type selector field (in Bytes) that will be put in front of the Union in the SOME/IP message.
stringEncoding	BaseTypeEncodingString	0..1	attr	Configures the encoding for SOME/IP serialization for the referenced dataPrototype in case of an String.

Table 3.101: ApSomeipTransformationProps

[TPS_MANI_03102]{DRAFT} **UserDefined serialization** ⌈ The UserDefinedTransformationProps meta-class that is referenced by the TransformationPropsToServiceInterfaceElementMapping in the role transformationProps provides the ability to define a User defined serialization for ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋ *(RS_MANI_00014, RS_MANI_00025)*

Please note that UserDefinedTransformationProps is derived from meta-class Identifiable and therefore has the ability to describe special data (sdg) by which it is possible to define custom structural extensions of an AUTOSAR model in a generic way. For more information about special data please refer to [6].

Class	TransformationPropsToServiceInterfaceElementMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties			
Note	Collection of TransformationPropsToServiceInterfaceElementMappings. **Tags:** atp.Status=draft atp.recommendedPackage=TransformationPropsToServiceInterfaceMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
mapping	TransformationPropsToServiceInterfaceElementMapping	*	aggr	Mapping that assigns serialization properties to elements of a ServiceInterface. **Tags:** atp.Status=draft

Table 3.102: TransformationPropsToServiceInterfaceElementMappingSet

Class	TransformationPropsToServiceInterfaceElementMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to associate a ServiceInterface element with TransformationProps. The referenced elements of the Service Interface will be serialized according to the settings defined in the TransformationProps. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
event	VariableDataPrototype	*	ref	This represents the reference to one or several events of one ServiceInterface. **Tags:** atp.Status=draft
field	Field	*	ref	This represents the reference to one or several fields of one ServiceInterface. **Tags:** atp.Status=draft
method	ClientServerOperation	*	ref	This represents the reference to one or several methods of one ServiceInterface. **Tags:** atp.Status=draft
tlvDataId	TlvDataIdDefinition	*	aggr	This aggregation represents the collection of tlvDataIds defined in the enclosing context. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=tlvDataId atp.Status=draft
transformationProps	TransformationProps	0..1	ref	This represents the reference to the applicable Serialization properties. **Tags:** atp.Status=draft

Table 3.103: TransformationPropsToServiceInterfaceElementMapping

Class	UserDefinedTransformationProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	The class UserDefinedTransformationProps specifies specific configuration properties of a user defined serializer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	UserDefinedTransformationProps			
–	–	–	–	–

Table 3.104: UserDefinedTransformationProps

3.15.2 Individual Definition of Serialization Properties

[TPS_MANI_03109]{DRAFT} **TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface** ⌈ The fine granular modeling of TransformationProps on the level of DataPrototypes overwrites the TransformationProps settings defined on the level of a ServiceInterface described with the TransformationPropsToServiceInterfaceElementMappingSet. ⌋*()*

[constr_3361]{DRAFT} **Selective definition of serialization settings** ⌈ If a SomeipDataPrototypeTransformationProps is defined for a composite DataPrototype of an element of a ServiceInterface (method, field, event) and if the reference someipTransformationProps exists then SomeipDataPrototypeTransformationProps that define the reference someipTransformationProps shall be defined for all other composite DataPrototypes of the ServiceInterface element as well. ⌋*()*

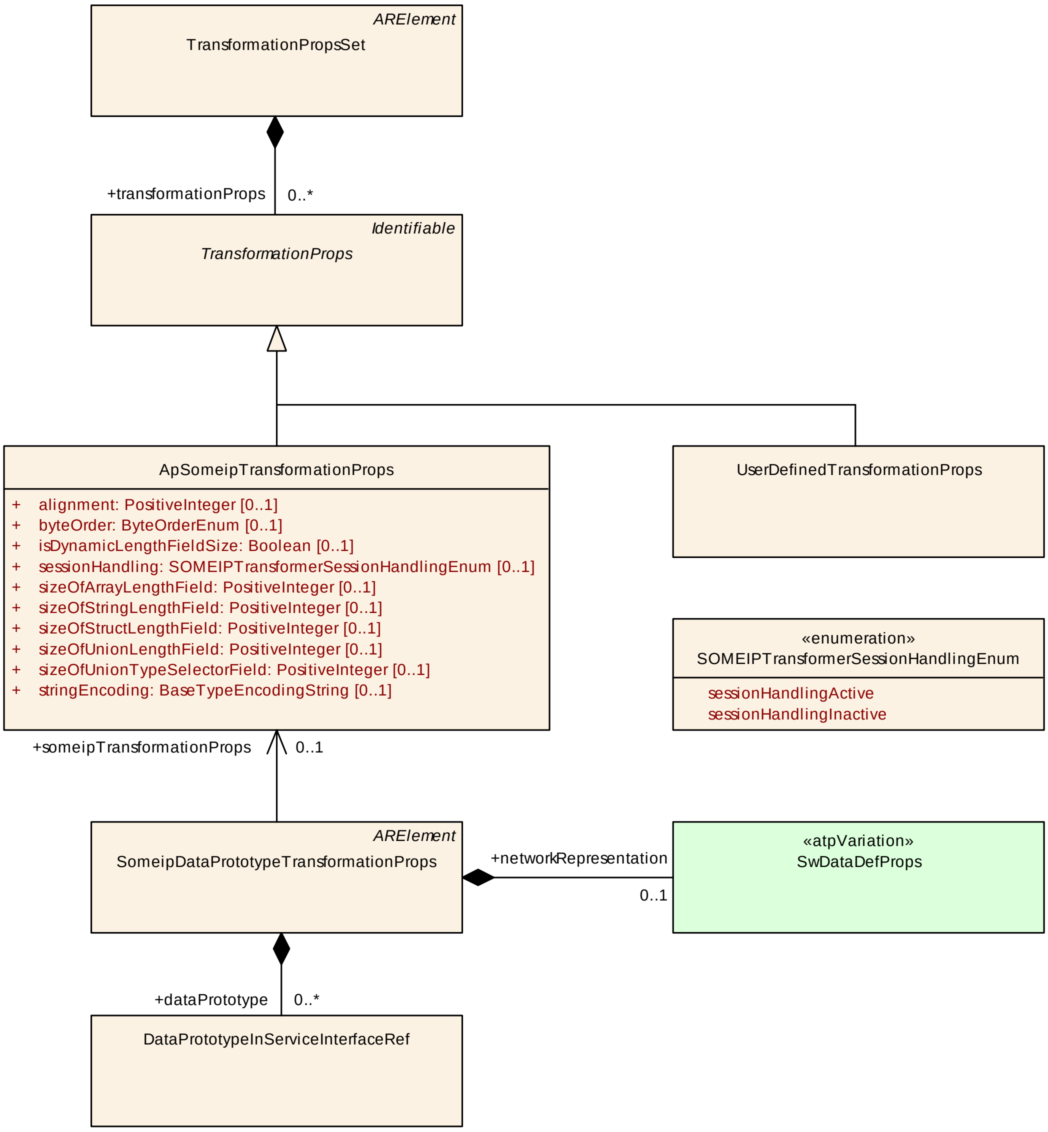


Figure 3.61: Overview about SOME/IP Serialization Properties

Class	**TransformationPropsSet**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	Collection of TransformationProps. **Tags:** atp.recommendedPackage=TransformationPropsSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
transformation Props	TransformationProps	*	aggr	Transformer specific configuration properties.

Table 3.105: TransformationPropsSet

Enumeration	SOMEIPTransformerSessionHandlingEnum
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer
Note	Enables or disable session handling for SOME/IP transformer
Literal	Description
sessionHandling Active	The SOME/IP Transformer shall use session handling **Tags:** atp.EnumerationValue=0
sessionHandling Inactive	The SOME/IP Transformer doesn't use session handling **Tags:** atp.EnumerationValue=1

Table 3.106: SOMEIPTransformerSessionHandlingEnum

[TPS_MANI_03070]{DRAFT} **Size of a length field for a chosen array or map** ⌈ The attribute sizeOfArrayLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a variable size array (vector), fixed size array or associative_map for which the SomeipDataPrototypeTransformationProps is defined, i.e. the variable size array (vector), fixed size array or associative_map that is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3353]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLengthField** ⌈ The value of the attribute sizeOfArrayLengthField shall be either 0, 1, 2 or 4. ⌋*()*

[constr_3447]{DRAFT} **ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0** ⌈ The sizeOfArrayLengthField value of 0 is only allowed to be used if a fixed size array for which the SomeipDataPrototypeTransformationProps is defined is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*()*

The setting of sizeOfArrayLengthField for fixed size arrays supports a backward compatible extension of such arrays with additional array elements.

[TPS_MANI_03071]{DRAFT} **Size of a length field for a chosen structure** ⌈ The attribute sizeOfStructLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a structure for which the SomeipDataPrototypeTransformationProps is defined, i.e. the structure that is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3354]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfStructLengthField** ⌈ The value of the attribute sizeOfStructLengthField shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03116]{DRAFT} **Size of a length field for a chosen string** ⌈ The attribute sizeOfStringLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a String for which the SomeipDataPrototypeTransformationProps is defined, i.e. the String that is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3372]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfStringLengthField** ⌈ The value of the attribute sizeOfStringLengthField shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03217]{DRAFT} **On-the-wire encoding for a chosen string** ⌈ The attribute stringEncoding of ApSomeipTransformationProps defines the on-the-wire encoding of a String for which the SomeipDataPrototypeTransformationProps is defined, i.e. the String that is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[TPS_MANI_03072]{DRAFT} **Size of a length field for a chosen union** ⌈ The attribute sizeOfUnionLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a union for which the SomeipDataPrototypeTransformationProps is defined, i.e. the union that is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3355]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLengthField** ⌈ The value of the attribute sizeOfUnionLengthField shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03073]{DRAFT} **Alignment of a dynamic DataPrototype** ⌈ The attribute alignment of ApSomeipTransformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of the variable data length data element for which the SomeipDataPrototypeTransformationProps is defined, i.e. the variable data length DataPrototype that is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3356]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.alignment** ⌈ The value of the attribute alignment shall always be divisible by 8. ⌋*()*

[TPS_MANI_03074]{DRAFT} **Size of a type selector field for a chosen union** ⌈ The attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps defines the size of a type selector field generated by SOME/IP in front of a union for which the SomeipDataPrototypeTransformationProps is defined, i.e. the union that is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3357]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField** ⌈ The value of the attribute sizeOfUnionTypeSelectorField shall be either 1, 2 or 4. ⌋*()*

[TPS_MANI_03075]{DRAFT} **Byte Order of chosen DataPrototype in the serialized data stream** ⌈ The attribute byteOrder of ApSomeipTransformationProps defines the Byte Order in front of the DataPrototype in the serialized data stream for which the SomeipDataPrototypeTransformationProps is defined, i.e. the DataPrototype that is referenced within the aggregated DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

Class	SomeipDataPrototypeTransformationProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties			
Note	This meta-class represents the ability to define data transformation props specifically for a SOME/IP serialization for a given DataPrototype. **Tags:** atp.Status=draft atp.recommendedPackage=SomeipDataPrototypeTransformationPropss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataPrototype	DataPrototypeInService InterfaceRef	*	aggr	Collection of DataPrototypes for which the settings in SomeipDataPrototypeTransformationProps are valid. For reuse reasons the SomeipDataPrototypeTransformation Props is able to aggregate several DataPrototypes. **Tags:** atp.Status=draft
network Representation	SwDataDefProps	0..1	aggr	Optional specification of the actual network representation for the referenced primitive DataPrototype. If a network representation is provided then the baseType available in the SwDataDefProps shall be used as input for the serialization/deserialization. If the network Representation is not provided then the baseType of the AbstractImplementationDataType shall be used for the serialization/deserialization. **Tags:** atp.Status=draft
someip Transformation Props	ApSomeip TransformationProps	0..1	ref	This reference represents the ability to define data transformation props specifically for a SOME/IP serialization. **Tags:** atp.Status=draft

Table 3.107: SomeipDataPrototypeTransformationProps

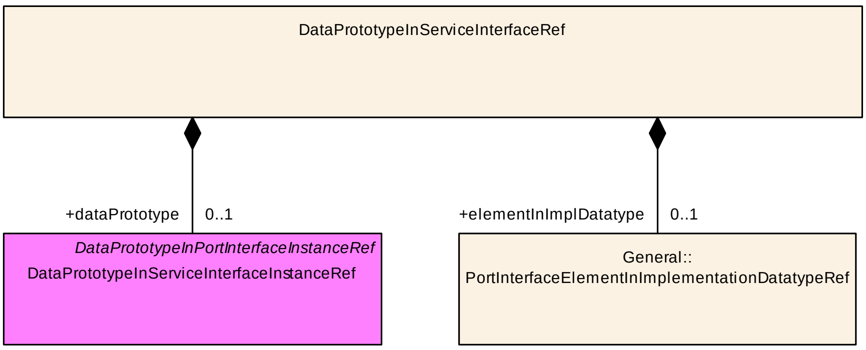


Figure 3.62: Reference to a DataPrototype in the context of a PortInterface that is typed by an ApplicationDataType or by a CppImplementationDataType

[TPS_MANI_01136]{DRAFT} **`AutosarDataPrototype` is the target of the `DataPrototypeInServiceInterfaceRef`** ⌈ If the target of an `DataPrototypeInServiceInterfaceRef` is an `AutosarDataPrototype` the role `DataPrototypeInServiceInterfaceRef.dataPrototype` shall be used to describe the reference **independently** of whether the `AutosarDataPrototype` is typed by an `ApplicationDataType` or a `CppImplementationDataType` and even **independently** of whether the `AutosarDataType` of the `AutosarDataPrototype` represents a composite data type. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[TPS_MANI_01137]{DRAFT} **Applicable use cases for DataPrototypeInServiceInterfaceRef** ⌈ Table 3.108 contains a comprehensive list of use cases for the usage of DataPrototypeInServiceInterfaceRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

Use case	Role
AutosarDataPrototype typed by an ApplicationDataType	dataPrototype
DataPrototype in AutosarDataPrototype typed by an ApplicationCompositeDataType	dataPrototype
AutosarDataPrototype typed by a CppImplementationDataType	dataPrototype
DataPrototype in AutosarDataPrototype typed by a CppImplementationDataType	elementInImplDatatype

Table 3.108: Possible use cases for the usage of DataPrototypeInServiceInterfaceRef

From a careful observation of Table 3.108 it should be clear that there is no valid use case to simultaneously use the two roles dataPrototype and elementInImplDatatype in the context of the same DataPrototypeInServiceInterfaceRef.

[constr_1551]{DRAFT} **Existence of DataPrototypeInServiceInterfaceRef.dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype** ⌈ For every given DataPrototypeInServiceInterfaceRef, either the aggregation DataPrototypeInServiceInterfaceRef.dataPrototype or DataPrototypeInServiceInterfaceRef.elementInImplDatatype shall exist. ⌋*()*

The usage of the SomeipDataPrototypeTransformationProps.networkRepresentation is explained in more detail in the System Template [16] in [TPS_SYST_02136] and [TPS_SYST_02137].

[constr_1615]{DRAFT} **Existence of attribute SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling** ⌈ The attribute SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling shall only exist if

- **none** of the aggregated dataPrototype.dataPrototype references
  - an ArgumentDataPrototype in the role DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype or
  - a DataPrototype referenced in the role DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype if the reference in the role contextDataPrototype exist in the enclosing DataPrototypeInServiceInterfaceInstanceRef.
- **none** of the aggregated aggregated dataPrototype.elementInImplDatatype references a DataPrototype referenced in the role PortInterfaceElementInImplementationDatatypeRef.targetDataPrototype.

⌋()

3.15.3 Assignment of TLV properties

3.15.3.1 Assignment of TLV Data IDs

[TPS_MANI_01097]{DRAFT} **Assignment of TLV data ids** ⌈ The assignment of TLV data ids is done in the context of the specification of TransformationPropsToServiceInterfaceElementMapping, namely by means of the attribute TransformationPropsToServiceInterfaceElementMapping.tlvDataId.id. ⌋ *(RS_MANI_00030)*

This approach takes benefit from the fact that the TlvDataIdDefinition is able to create references to relevant model elements.

The assignment of the TLV data id is therefore done by creating such a reference and assigning a TLV data id to it by means of the attribute TlvDataIdDefinition.id.

Please note that the assignment of TLV data ids is compulsory for an entire data structure that has at least one optional member. In a nutshell, this conclusion (that is also backed by [PRS_SOMEIP_00230], see [8]) is the motivation for the existence of [constr_1594], and [constr_1595].

Please note further that the assignment of TLV data ids is not restricted to data structures with optional members. There is also a use case to support sending the elements of a specific data structure in arbitrary order even if none of the elements is considered optional.

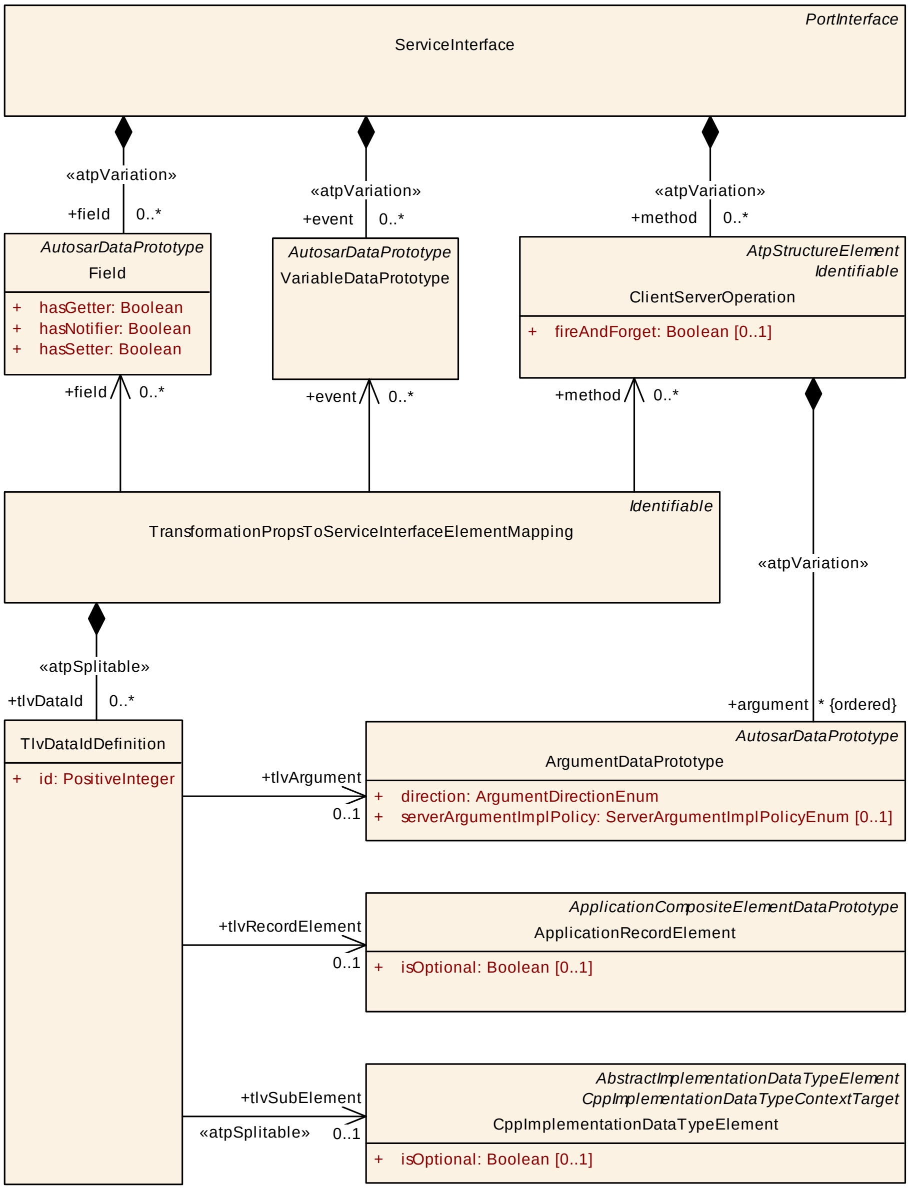


Figure 3.63: Modeling of the TLV data id

[constr_1594]{DRAFT} **Consistent assignment of TLV data ids to ApplicationRecordDataType** ⌈ For every ApplicationRecordDataType where direct members set the attribute ApplicationRecordElement.isOptional to the value True references to **all direct members** of this ApplicationRecordDataType shall be created on the basis of the definition of TlvDataIdDefinition. ⌋*()*

[constr_1595]{DRAFT} **Consistent assignment of TLV data ids to CppImplementationDataType or CppImplementationDataTypeElement** ⌈ For every CppImplementationDataType of category STRUCTURE where direct members set the attribute CppImplementationDataTypeElement.isOptional to the value True references to **all direct members** of this CppImplementationDataType shall be created on the basis of the definition of TlvDataIdDefinition. ⌋*()*

The definition of a TlvDataIdDefinition that refers to an eligible model element is not limited to scenarios where optional elements are defined. It is also possible to define TlvDataIdDefinition for arbitrary methods or data structures.

A typical use case could be to prepare the argument list or sub-elements for future extensions. However, if one argument or sub-element is referenced then it is necessary to define references from TlvDataIdDefinitions to all other arguments or sub-elements as well.

[constr_1593]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvArguments** ⌈ If the reference TlvDataIdDefinition.tlvArgument exists for one argument of a given ClientServerOperation then further TlvDataIdDefinition.tlvArgument shall exist **for all** arguments of the given ClientServerOperation and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping. ⌋*()*

[constr_1603]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvRecordElements** ⌈ If the reference TlvDataIdDefinition.tlvRecordElement exists for one element of a given ApplicationRecordDataType then further TlvDataIdDefinition.tlvRecordElement shall exist **for all** elements of the given ApplicationRecordDataType and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping. ⌋*()*

[constr_1604]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvSubElements** ⌈ If the reference TlvDataIdDefinition.tlvSubElement exists for one subElement of a given CppImplementationDataType or CppImplementationDataTypeElement then further TlvDataIdDefinition.tlvSubElement shall exist **for all** subElements of the given CppImplementationDataType or CppImplementationDataTypeElement and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping. ⌋*()*

Class	**TlvDataIdDefinition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties			
Note	This meta-class represents the ability to define the tlvDataId. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
id	PositiveInteger	1	attr	This attribute represents the definition of the value of the TlvDataId
tlvArgument	ArgumentDataPrototype	0..1	ref	This reference assigns a tlvDataId to a given argument of a ClientServerOperation. **Tags:** atp.Status=draft

▽

△

Class	TlvDataIdDefinition			
tlvRecord Element	ApplicationRecord Element	0..1	ref	This reference associates the definition of a TLV data id with a given ApplicationRecordElement. **Tags:** atp.Status=draft
tlvSubElement	CppImplementation DataTypeElement	0..1	ref	This reference associates the definition of a TLV data id with a given CppImplementationDataTypeElement. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=tlvSubElement atp.Status=draft

Table 3.109: TlvDataIdDefinition

The definition of a TlvDataIdDefinition.id has the purpose to provide means to unambiguously identify the argument or sub-element. For this purpose, the value of the id needs to be unique in the respective context.

[constr_1596]{DRAFT} **Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ArgumentDataPrototype** ⌈ For all TlvDataIdDefinition that are referencing ArgumentDataPrototypes of a given ClientServerOperation in the role tlvArgument, the attribute TlvDataIdDefinition.id **shall exist and have a unique value per communication direction**, i.e. in the context of the collection of all

- arguments where attribute direction is set to either in or inout
- arguments where attribute direction is set to either out or inout
- arguments where attribute direction is set to inout (if the method **only** has arguments where attribute direction is set to inout)

of the respective enclosing ClientServerOperation. ⌋*()*

Rationale for the existence of [constr_1596]: arguments where attribute direction is set to either in or inout are never sent in the same SOME/IP message as arguments where attribute direction is set to either out or inout.

[constr_1597]{DRAFT} **Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ApplicationRecordElement** ⌈ For all TlvDataIdDefinition that are referencing ApplicationRecordElements of a given ApplicationDataType in the role tlvRecordElement the attribute TlvDataIdDefinition.id **shall exist and have a unique value** in the context of respective enclosing ApplicationRecordDataType. ⌋*()*

[constr_1598]{DRAFT} **Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement** ⌈ For all TlvDataIdDefinition that are referencing CppImplementationDataTypeElements of a given CppImplementationDataType/CppImplementationDataTypeElement in the role tlvSubElement the attribute TlvDataIdDefinition.id **shall exist and have a unique value** in the context of respective enclosing CppImplementationDataType or CppImplementationDataTypeElement. ⌋*()*

Obviously, it is necessary to avoid ambiguity with respect to the definition of TLV data ids. Each model element that can be assigned such an id shall only be assigned one id.

[constr_1599]{DRAFT} **TlvDataIdDefinition referencing ArgumentDataPrototype** ⌈ Each ArgumentDataPrototype shall be referenced **at most once** in the role tlvArgument in the context of the same TransformationPropsToServiceInterfaceElementMapping. ⌋*()*

[constr_1600]{DRAFT} **TlvDataIdDefinition referencing ApplicationRecordElement** ⌈ Each ApplicationRecordElement shall be referenced **at most once** in the role tlvRecordElement in the context of the same TransformationPropsToServiceInterfaceElementMapping. ⌋*()*

[constr_1601]{DRAFT} **TlvDataIdDefinition referencing CppImplementationDataTypeElement** ⌈ Each CppImplementationDataTypeElement shall be referenced **at most once** in the role tlvSubElement in the context of the same TransformationPropsToServiceInterfaceElementMapping. ⌋*()*

[constr_1628]{DRAFT} **Definition of static length field sizes in case of TLV usage** ⌈ If the aggregation tlvDataId exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,
- sizeOfStringLengthField,
- sizeOfStructLengthField, and
- sizeOfUnionLengthField

shall have a value greater than 0. ⌋*()*

Rationale for the existence of [constr_1628]: The TLV serialization requires the usage of length fields:

- If wire type 4 is used (for more details, please refer to [TPS_MANI_01186]) then the length field size shall be statically configured.
- If wire types 5-7 are used (see [TPS_MANI_01186]) then the static configuration of the length field size shall also be present since not all length fields are preceded by a tag, e.g. structures contained in an array or the top-level structure contained in a SOME/IP event.

Without demanding the existence of length fields in such a case the result of a serialization could be ambiguous, i.e. make it impossible for the de-serializer to figure out the data layout[8].

[8] If a structure consists only of optional elements it would be hard to detect the case where an array element carries such a structure that happens to set all elements to non-available.

[constr_1629]{DRAFT} **Identical sizes of length fields in case of TLV usage** ⌈ If the aggregation tlvDataId exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,
- sizeOfStringLengthField,
- sizeOfStructLengthField, and
- sizeOfUnionLengthField

shall have an identical value. ⌋*()*

Rationale for the existence of [constr_1629]: if wire type 4 is used (for more details, please refer to [TPS_MANI_01186]) and if the receiver encounters a member of a structure or an argument with an unknown tag the de-serializer cannot determine the actual data type of the member of the structure or argument.

[constr_1630]{DRAFT} **No definition of length field sizes on DataPrototype level in case of TLV usage** ⌈ If the aggregation tlvDataId exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,
- sizeOfStringLengthField,
- sizeOfStructLengthField, and
- sizeOfUnionLengthField

shall not be individually defined on the level of a DataPrototype (i.e. by means of the reference SomeipDataPrototypeTransformationProps.someipTransformationProps) but only on the level of a ServiceInterface (i.e. by means of the reference TransformationPropsToServiceInterfaceElementMapping.transformationProps. ⌋*()*

Rationale for the existence of [constr_1630]: if wire type 4 is used (for more details, please refer to [TPS_MANI_01186]) and if the receiver encounters a member or argument with an unknown tag the de-serializer needs to know the size of the length field.

The most reliable way to achieve this is to demand the definition of the size of the length field on the level of the ServiceInterface.

3.15.3.2 Assignment of Wire Type Selection

The TLV encoding supports the definition of a so-called wire type that controls how the information about the length of length fields shall be interpreted.

The meaning of specific settings of the wire type is defined in [8, PRS SOME/IP Protocol].

[TPS_MANI_01186]{DRAFT} **Definition of the applicable wire type** ⌈ Attribute ApSomeipTransformationProps.isDynamicLengthFieldSize shall be used to define the applicable wire type.

If the value of attribute ApSomeipTransformationProps.isDynamicLengthFieldSize is set to True then wire type 5-7 shall be used.

If the value of attribute ApSomeipTransformationProps.isDynamicLengthFieldSize does not exist or is set to False then wire type 4 shall be used. ⌋ *(RS_MANI_00030)*

3.16 Process Design

Within the definition of e.g. a diagnostic mapping, the assignment to the Process is typically done in a methodological step[9] that happens when all the diagnostic mapping[10] is already complete.

Therefore, it would be good to implement a proxy for an actual Process that can stand in as the target of the relation to a Process at design time. This semantics is realized by meta-class ProcessDesign.

[TPS_MANI_01228]{DRAFT} **Semantics of meta-class ProcessDesign** ⌈ Meta-class ProcessDesign shall be used whenever a design-time representation is required for a Process that is designed in a **later** step in the workflow as part of the deployment specification. ⌋*()*

The integrator would have to take care that an actual Process refers to the corresponding ProcessDesign such that by means of this reference an AUTOSAR software tool is able to figure out the relation between a diagnostic mapping and a process, provided that each ProcessDesign is **only** referenced by a single Process.

[constr_1550]{DRAFT} **Reference from Process to ProcessDesign** ⌈ Each ProcessDesign shall only be referenced from a single Process. ⌋*()*

Note that the reference from the Process to the ProcessDesign acknowledges the fact that the Process is typically created later in time[11].

[9] i.e. during the creation of the execution manifest
[10] From the methodological point of view, the creation of the diagnostic mapping is typically considered a design-time activity.
[11] In other words, if references are needed between design-related and deployment-related meta-classes then the direction of these references shall always point from deployment to design.

Class	ProcessDesign			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign			
Note	This meta-class has the ability to stand in for a Process at the time when the Process does not yet exist. But its future existence already needs to be considered during design phase and for that a dedicated model element is required.. **Tags:** atp.Status=draft atp.recommendedPackage=ProcessDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
deterministic ClientResource Needs	DeterministicClient ResourceNeeds	*	aggr	This aggregation represents the collection of applicable resource needs for the design of deterministic clients. **Tags:** atp.Status=draft
executable	Executable	0..1	ref	Reference to executable that is executed in the process. **Tags:** atp.Status=draft

Table 3.110: ProcessDesign

Conceivably, the association of diagnostic mappings with Meta-class ProcessDesign may still happen as a finalizing last step of the activity to create the diagnostic mappings. To accommodate for this potential modeling, the reference from a diagnostic mapping to ProcessDesign has been decorated by stereotype ≪atpSplitable≫.

For more information concerning the semantics of this stereotype please refer to the specification of the AUTOSAR Generic Structure Template [6].

[constr_1693]{DRAFT} **Relation of Executable, ProcessDesign, and Process** ⌈ Any Executable that is referenced by a ProcessDesign shall also be referenced by every Process that references the ProcessDesign. ⌋*()*

[TPS_MANI_01229]{DRAFT} **Pre-allocation of a given ProcessDesign on a specific MachineDesign** ⌈ It is also possible to pre-allocate a given ProcessDesign on a specific MachineDesign. For this purpose meta-class ProcessDesignToMachineDesignMapping exists. ⌋*()*

The semantics of meta-class MachineDesign is explained in section 5.

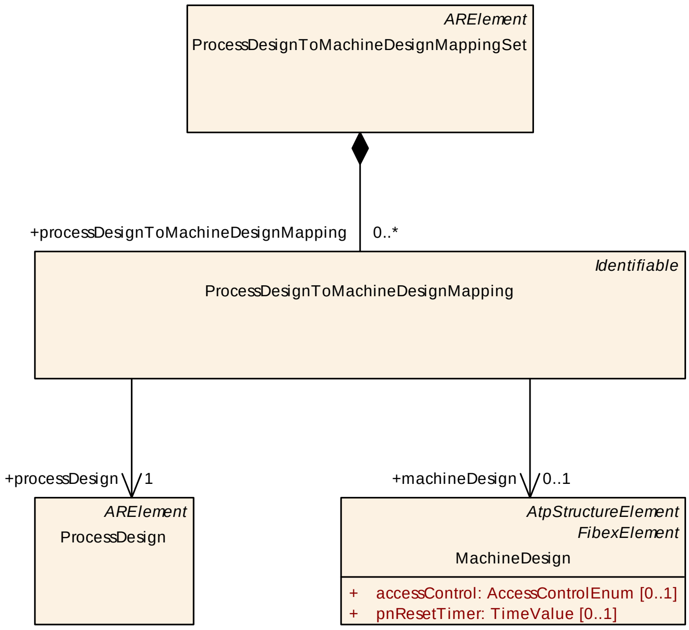


Figure 3.64: Modeling of the ProcessDesignToMachineDesignMapping

Class	**ProcessDesignToMachineDesignMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This element is used in the design phase to predefine a mapping of a process to a machine. Such a mapping may be overruled in the deployment phase. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
machineDesign	MachineDesign	0..1	ref	This reference identifies the MachineDesign in the context of the ProcessDesignToMachineDesignMapping. **Tags:** atp.Status=draft
processDesign	ProcessDesign	1	ref	This reference identifies the ProcessDesign in the context of the ProcessDesignToMachineDesignMapping. **Tags:** atp.Status=draft

Table 3.111: ProcessDesignToMachineDesignMapping

Please note that an intended `ProcessDesignToMachineDesignMapping` may not be possible for utilization of the target machine and therefore a different `ProcessToMachineMapping` may be created in the deployment phase.

3.16.1 Deterministic Client Resource

Meta-class `ProcessDesign` can also be used to add support for the so-called `Deterministic Client`.

Please note that an explanation of the specific meaning of the term `Deterministic Client` is out of the scope of this document. A detailed explanation can be found in the SWS Execution Management [17].

To formalize the support for the Deterministic Client, meta-class DeterministicClientResourceNeeds is aggregated at ProcessDesign.

Class	**DeterministicClientResourceNeeds**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign			
Note	This meta-class specifies process and cycle specific computing resource needs of DeterministicClient library functions. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hardware Platform	String	0..1	attr	This attribute represents a textual identification of the target platform.
initResource	DeterministicClient Resource	0..1	aggr	This represents the computing resource needs of a DeterministicClient::WaitForNextActivation kInit cycle. **Tags:** atp.Status=draft
runResource	DeterministicClient Resource	0..1	aggr	This represents the computing resource needs of a DeterministicClient::WaitForNextActivation kRun cycle. **Tags:** atp.Status=draft

Table 3.112: DeterministicClientResourceNeeds

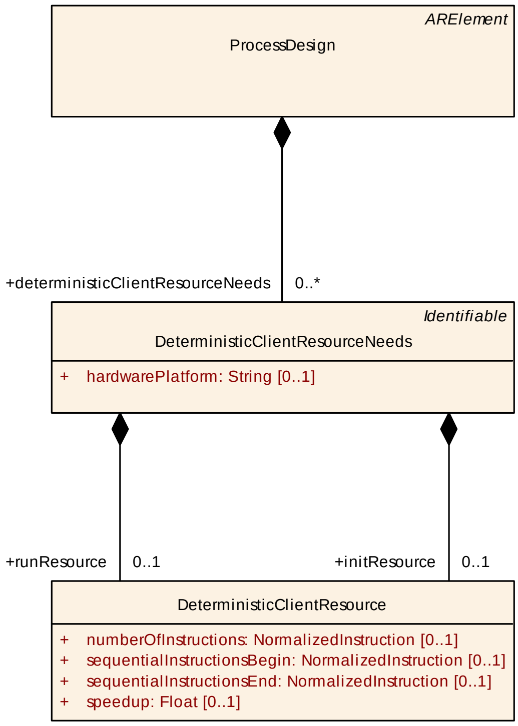


Figure 3.65: Modeling of the DeterministicClientResourceNeeds

[TPS_MANI_01199]{DRAFT} **Semantics of DeterministicClientResourceNeeds** ⌈ Meta-class DeterministicClientResourceNeeds aggregates DeterministicClientResource in two roles in order to be able to specify resource needs in two different contexts of the execution of a Deterministic Client. ⌋ *(RS_MANI_00050)*

Class	DeterministicClientResource			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign			
Note	This meta-class specifies computing resource needs of DeterministicClient library functions. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
numberOf Instructions	NormalizedInstruction	0..1	attr	This attribute represents the normalized runtime consumption on the target system within one DeterministicClient::WaitForNextActivation cycle, assuming the "worst-case" runtime where the workers would be executed sequentially.
sequential Instructions Begin	NormalizedInstruction	0..1	attr	Normalized sequential runtime at the beginning of the DeterministicClient::WaitForNextActivation cycle (which mostly cannot be parallelized), before the main usage of the worker pool starts.
sequential InstructionsEnd	NormalizedInstruction	0..1	attr	WaitForNextActivation cycle (which mostly cannot be parallelized), after the main usage of the worker pool has ended.
speedup	Float	0..1	attr	This attribute defines how much faster the calculations within one DeterministicClient::WaitForNextActivation cycle can be finished if numberOfWorkers are physically available, i.e. if enough cores were available on the machine to perform parallel execution of all workers (sequential runtime / parallelized runtime).

Table 3.113: DeterministicClientResource

[TPS_MANI_01200]{DRAFT} **Semantics of meta-class DeterministicClientResource** ⌈ Meta-class DeterministicClientResource defines several attributes that provide information about the nature of the execution of worker threads. The values of these attributes are given a dimensionless NormalizedInstruction.

Nevertheless, the values of the attributes

- numberOfInstructions
- sequentialInstructionsBegin
- sequentialInstructionsEnd

are only valid for a specific hardware platform. The purpose of using NormalizedInstruction is to align resource usage of different Processes (possibly from different vendors) at integration time. ⌋*(RS_MANI_00050)*

Primitive	NormalizedInstruction
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign
Note	This meta-class is used to describe runtime budget needs on the target system within Deterministic Client::WaitForNextActivation cycles. NormalizedInstructions does not reflect the actual number of code instructions, but allows the description of comparative resource needs. NormalizedInstructions is used for configuration of computing resources at integration time. NormalizedInstruction = runtime in sec * clock frequency in Hz ▽

▽

△

Primitive	**NormalizedInstruction**
	△ **Tags:** atp.Status=draft xml.xsd.customType=NORMALIZED-INSTRUCTION xml.xsd.pattern=[1-9][0-9]* xml.xsd.type=string

Table 3.114: NormalizedInstruction

3.17 Grant Design

The definition of capabilities (for example: `ClientComSpec.clientCapability` as described in chapter 3.12.5.1.3 is used to express the intention of the software designer to use (or refrain from using) specific APIs in the application software.

The definition of capabilities represents one aspect of the identity and Access Management (IAM). Another aspect of the IAM configuration is the definition of the actual permissions granted by the platform software.

The modeling of such grants is done on two levels:

- the definition of `GrantDesign` allows for the pre-specification of grants already on the design level. The modeling of `GrantDesign` is described in this chapter.
- the definition of `Grant` allows for the actual and final specification of grants from the perspective of the platform software. The modeling of Grant is described in chapter 8.9.

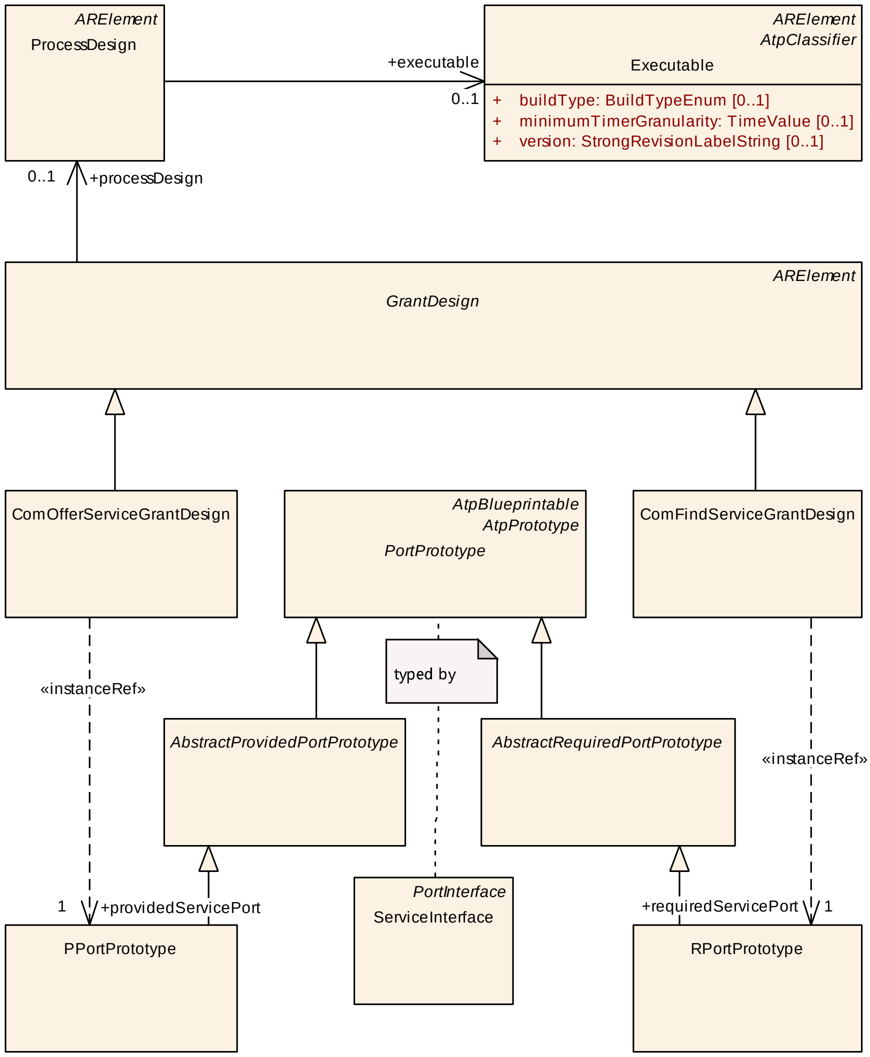


Figure 3.66: Modeling of grant designs for service discovery

Class	***GrantDesign*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign			
Note	This meta-class serves as an abstract base class for the description of grants on design level. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ComEventGrantDesign, ComFieldGrantDesign, ComFindServiceGrantDesign, ComMethodGrantDesign, ComOfferServiceGrantDesign			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
processDesign	ProcessDesign	0..1	ref	This reference identifies the corresponding Process Design that gives context to the GrantDesing. **Tags:** atp.Status=draft

Table 3.115: GrantDesign

Abstract meta-class `GrantDesign` acts as the base class for the definition of grants on the design level.

Grants are specific for a given Process. In other words, two Processes created from the same Executable may be assigned different sets of grants. This specific relation shall also be available on the design level.

[TPS_MANI_01231]{DRAFT} **GrantDesign references ProcessDesign** ⌈ Meta-class GrantDesign references ProcessDesign as a means to design the set of Grants for the given Process. ⌋*(RS_MANI_00060)*

3.17.1 Com Grant Design

Subclasses of GrantDesign are created to cover specific aspects of grants for communication on the *AUTOSAR adaptive Platform*.

[TPS_MANI_01232]{DRAFT} **Semantics of meta-class ComOfferServiceGrantDesign** ⌈ The existence of a ComOfferServiceGrantDesign that references a specific AbstractProvidedPortPrototype in the role providedServicePort indicates that the design foresees that the referenced AbstractProvidedPortPrototype shall be granted rights to offer the respective service. ⌋*(RS_MANI_00060)*

Please note that there is no explicitly modeled capability that corresponds to the existence of the ComOfferServiceGrantDesign. The understanding is that he mere existence of an AbstractProvidedPortPrototype typed by a ServiceInterface indicates the capability to offer a service.

Class	**ComOfferServiceGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for offering a service. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
providedService Port	PPortPrototype	1	iref	This instanceRef identifies the PPortPrototype on which the service shall be offered. **Tags:** atp.Status=draft

Table 3.116: ComOfferServiceGrantDesign

[TPS_MANI_01233]{DRAFT} **Semantics of meta-class ComFindServiceGrantDesign** ⌈ The existence of a ComFindServiceGrantDesign that references a specific AbstractRequiredPortPrototype in the role requiredServicePort indicates that the design foresees that the referenced AbstractRequiredPortPrototype shall be granted rights to find the respective service. ⌋*(RS_MANI_00060)*

Please note that there is no explicitly modeled capability that corresponds to the existence of the ComFindServiceGrantDesign. The understanding is that the mere existence of an AbstractRequiredPortPrototype typed by a ServiceInterface indicates the capability to find a service.

Class	**ComFindServiceGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for finding a service. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
requiredService Port	RPortPrototype	1	iref	This instanceRef identifies the RPortPrototype on which the service shall be found. **Tags:** atp.Status=draft

Table 3.117: ComFindServiceGrantDesign

[TPS_MANI_01234]{DRAFT} **Semantics of ComFieldGrantDesign** ⌈ The existence of a ComFieldGrantDesign that references a specific Field in the role field indicates that the design foresees that the application software shall be granted rights to access the respective Field. The nature of the access, i.e. get vs. set is specified by means of the attribute role. ⌋*(RS_MANI_00060)*

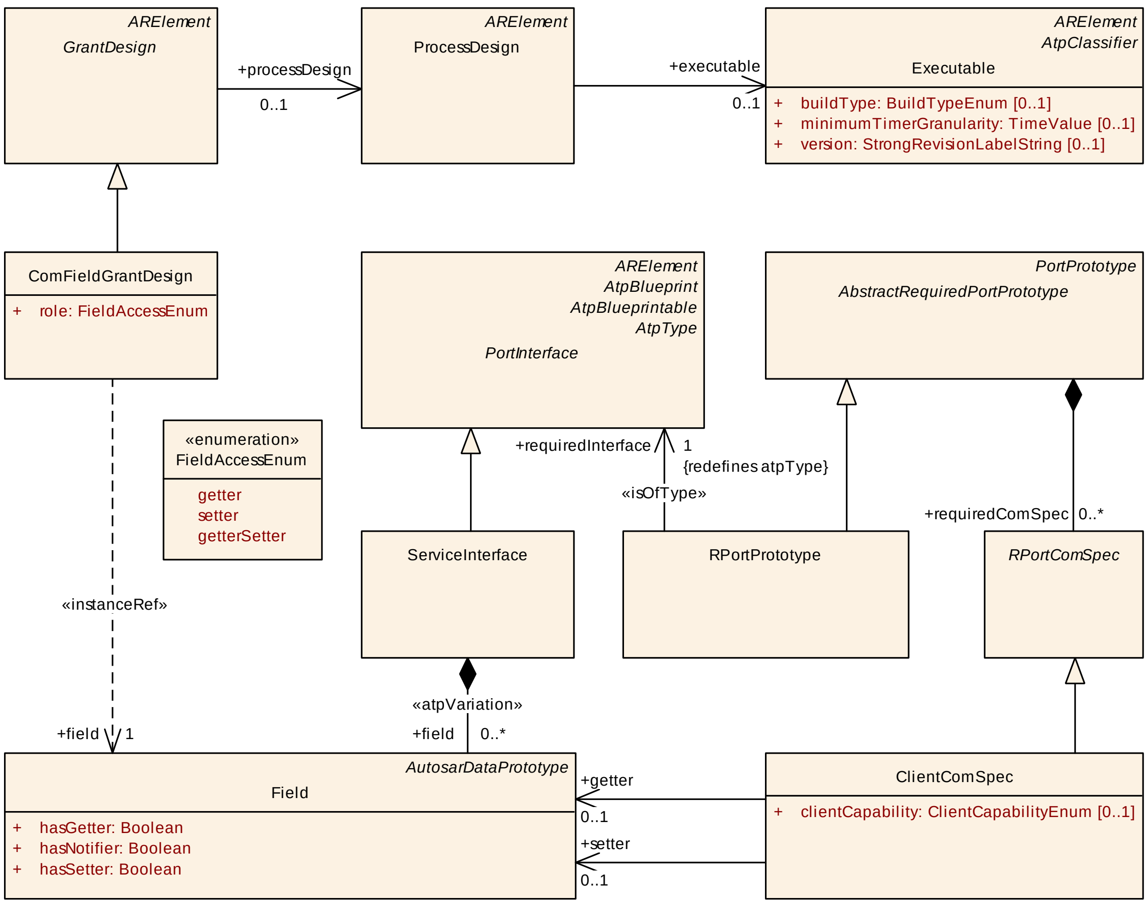


Figure 3.67: Modeling of grant designs for field

Class	**ComFieldGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.field. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
field	Field	1	iref	Reference to the affected Field in the context of an Executable. **Tags:** atp.Status=draft
role	FieldAccessEnum	1	attr	This attribute provides the ability to further specify the access to the ServiceInterface.field from a design perspective.

Table 3.118: ComFieldGrantDesign

Enumeration	**FieldAccessEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant
Note	This meta-class provides values that qualify access to a field. **Tags:** atp.Status=draft
Literal	*Description*
getter	Access to the getter of the Field. **Tags:** atp.EnumerationValue=0
getterSetter	Access to getter and setter of the field **Tags:** atp.EnumerationValue=2
setter	Access to the setter of the Field. **Tags:** atp.EnumerationValue=1

Table 3.119: FieldAccessEnum

[TPS_MANI_01235]{DRAFT} **Semantics of `ComEventGrantDesign`** ⌈ The existence of a `ComEventGrantDesign` that references a specific `VariableDataPrototype` that is aggregated in the role `event` by the enclosing `ServiceInterface` indicates that the design foresees that the application software shall be granted rights to access the respective `event`. ⌋*(RS_MANI_00060)*

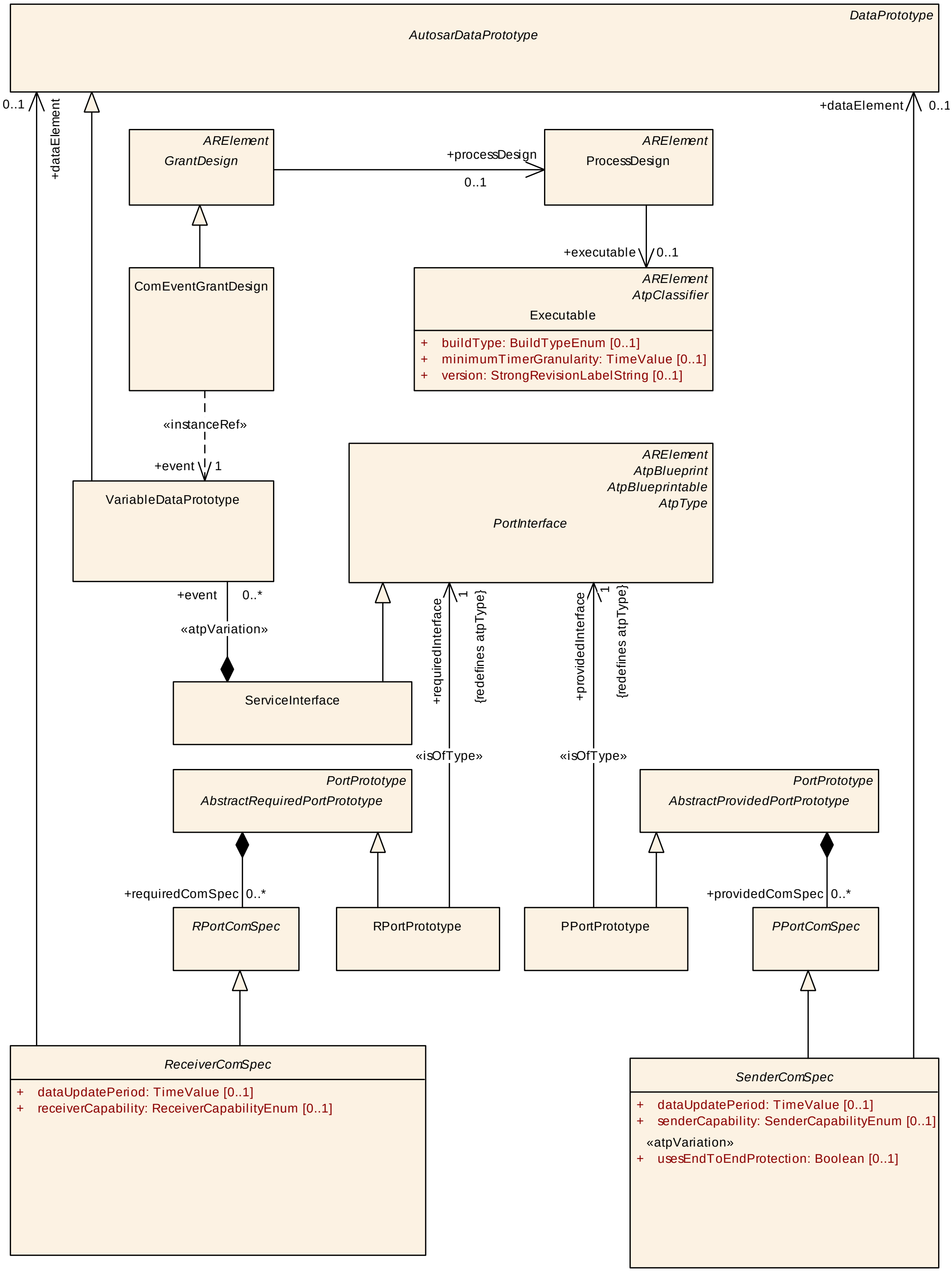


Figure 3.68: Modeling of grant designs for `event`

Class	ComEventGrantDesign			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.event. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
event	VariableDataPrototype	1	iref	This reference represents the affected event. **Tags:** atp.Status=draft

Table 3.120: ComEventGrantDesign

[TPS_MANI_01236]{DRAFT} **Semantics of ComMethodGrantDesign** ⌈ The existence of a ComMethodGrantDesign that references a specific ClientServerOperation that is aggregated in the role method by the enclosing ServiceInterface indicates that the design foresees that the application software shall be granted rights to call the respective method. ⌋*()*

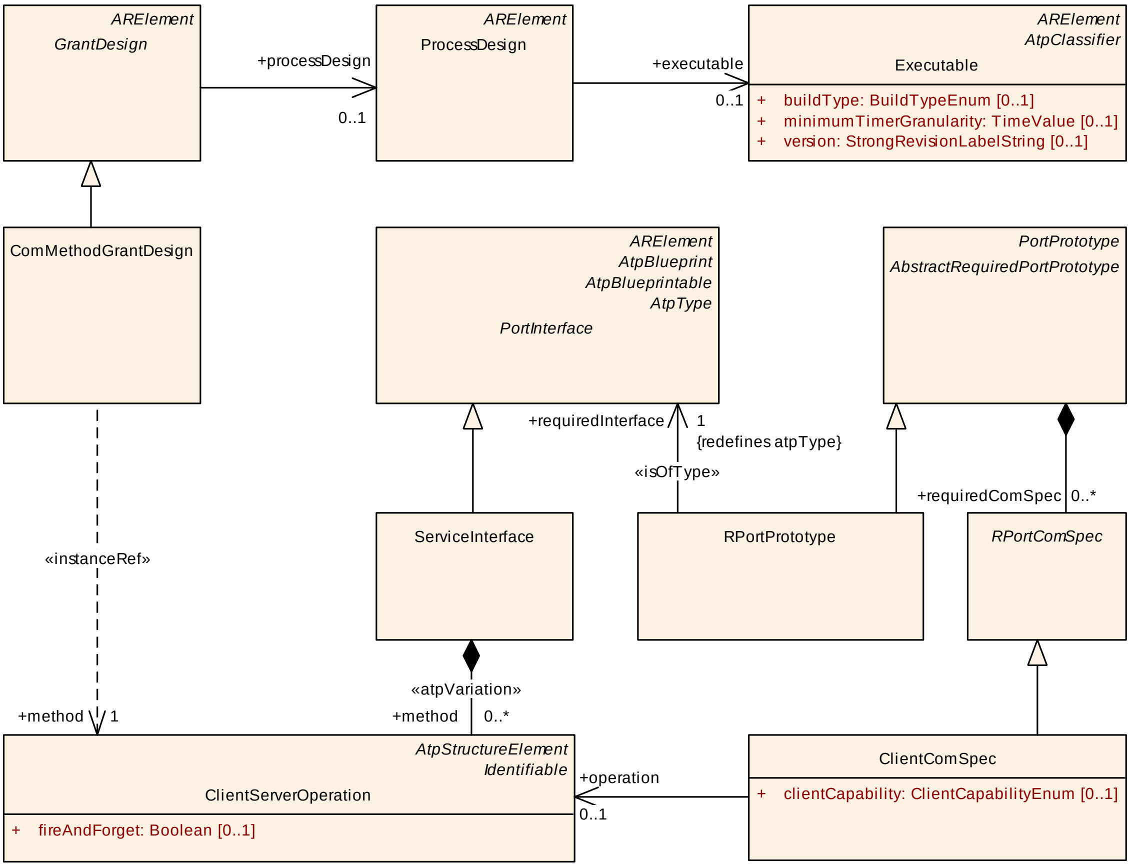


Figure 3.69: Modeling of grant designs for method

Class	**ComMethodGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.method. **Tags:** atp.Status=draft atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
method	ClientServerOperation	1	iref	This reference identifies the corresponding method. **Tags:** atp.Status=draft

Table 3.121: ComMethodGrantDesign

4 Diagnostic Design

4.1 Diagnostic Mapping

4.1.1 Overview

The configuration of diagnostics on the *AUTOSAR adaptive platform* will typically be done by creating a Diagnostic Extract by means of the Diagnostic Extract Template [18] that is also used on the *AUTOSAR classic platform*.

Therefore, concepts within the Diagnostic Extract should be similarly applicable to models on both platforms in a uniform fashion.

It can even be safely expected that a given Diagnostic Extract can be divided into parts that apply for ECUs build on top of the *AUTOSAR classic platform* and parts that apply to ECUs built on top of the *AUTOSAR adaptive platform* that all belong to the same vehicle.

In terms of applicability to this document, the part of the Diagnostic Extract that is relevant in this context is the mapping between the definition of information related to diagnostic protocol content and the application software.

Following the pattern of communication on the *AUTOSAR adaptive platform*, interaction between the application software and platform modules for diagnostics (the so-called `AUTOSAR Adaptive Diagnostic Management`) is also using service-oriented communication.

This raises the question how the communication ends on both application and platform software get together in the course of a service discovery. This issue can be addressed by utilizing modeling concepts existing in a Diagnostic Extract on the *AUTOSAR adaptive platform*.

Specifically, by formally modeling the relation between the `AUTOSAR Adaptive Diagnostic Management` and specific endpoints in the application software it is possible to configure the service-oriented communication in a way that communication endpoints that are supposed to be connected become actually connected to each other as the service discovery unfolds.

The meta-classes that need to be considered for this purpose are in the following list:

- `DiagnosticServiceDataMapping`
- `DiagnosticEventPortMapping`
- `DiagnosticOperationCyclePortMapping`
- `DiagnosticEnableConditionPortMapping`
- `DiagnosticStorageConditionPortMapping`
- `DiagnosticClearConditionPortMapping`

- DiagnosticIndicatorPortMapping
- DiagnosticMemoryDestinationPortMapping
- DiagnosticSecurityLevelPortMapping
- DiagnosticGenericUdsPortMapping
- DiagnosticServiceDataIdentifierPortMapping

In order to exemplify the approach, the diagram depicted in Figure 4.1 describes a very simplistic situation where two different PPortPrototypes typed by possibly two different DiagnosticDataIdentifierInterface exposed by an AdaptiveApplicationSwComponentType is accessed by the AUTOSAR Adaptive Diagnostic Management on the *AUTOSAR adaptive platform* with the purpose of accessing an entire DID.

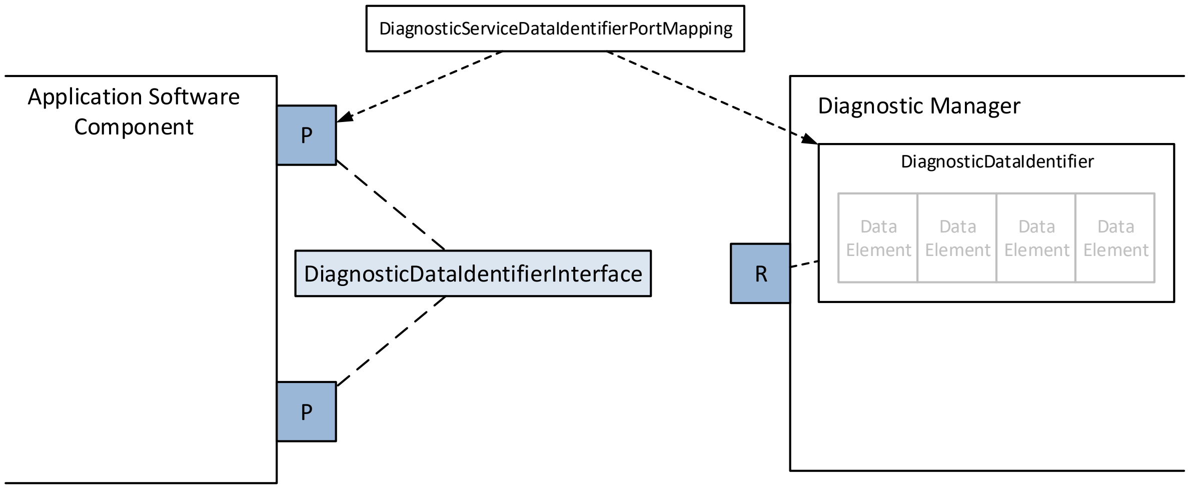


Figure 4.1: Example data exchange for diagnostic purpose

In this situation, the AUTOSAR Adaptive Diagnostic Management obviously needs to be aware which of the two available PPortPrototypes has to be accessed from the depicted RPortPrototype for working with a given DiagnosticDataIdentifier in particular.

If it were possible to identify the matching pairs of PortPrototypes then the communication channel between them could be established either completely or at lest ot a large extent automatically. Please note that this statement might or might not involve the execution of a service discovery. In many case it will not.

From the technical point of view, the AUTOSAR meta-model provides means to achieve the discussed formalization of the relation between an element of the diagnostics configuration (in this case a DiagnosticDataIdentifier that is represented by a PortPrototype on the surface of the DM) and a PortPrototype exposed by the application software.

In particular, a subclass of DiagnosticMapping (in this specific case: DiagnosticServiceDataIdentifierPortMapping) formalizes the “connection” between both ends of the communication.

Of course, the specifics of the PortPrototype on the side of the AUTOSAR Adaptive Diagnostic Management need to be derived from the configuration (in this case, the definition of a DiagnosticDataIdentifier) of the external behavior of the diagnostic stack on the *AUTOSAR adaptive platform*, as described by a corresponding Diagnostic Extract [18].

A further kind of mapping that is necessary to enable diagnostics on the *AUTOSAR adaptive platform* comes with slightly more complexity.

In this case use-cases are implemented that may or may not involve several communication ends (in the form of PortPrototypes).

Class	**DiagnosticDataIdentifier**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to model a diagnostic data identifier (DID) that is fully specified regarding the payload at configuration-time. **Tags:** atp.recommendedPackage=DiagnosticDataIdentifiers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticAbstractDataIdentifier*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataElement	DiagnosticParameter	1..*	aggr	This is the dataElement associated with the DiagnosticDataIdentifier. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataElement, variationPoint.shortLabel vh.latestBindingTime=postBuild
didSize	PositiveInteger	0..1	attr	This attribute indicates the size in bytes of the DiagnosticDataIdentifier.
representsVin	Boolean	0..1	attr	This attributes indicates whether the specific DiagnosticDataIdentifier represents the vehicle identification.
supportInfoByte	DiagnosticSupportInfoByte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier.

Table 4.1: DiagnosticDataIdentifier

The response to this situation on the *AUTOSAR classic platform* has been the definition of the SwcServiceDependency that allows for associating several PortPrototypes in specific roles to a given use-case.

Although the need for involving different PortPrototypes in the implementation of a given use case might slightly have gone down, there is still enough motivation to keep using this pattern on the *AUTOSAR adaptive platform* as well.

For example, one benefit of this approach over a seemingly more straightforward implementation to refer to a PortPrototype directly is the ability to let several PortPrototypes (where e.g. some may represent server functionality, and the rest could represent client functionality) in concert in order to implement a given use case.

Figure 4.2 provides a visual explanation of how this kind of diagnostic mapping to model elements on the *AUTOSAR adaptive platform* works.

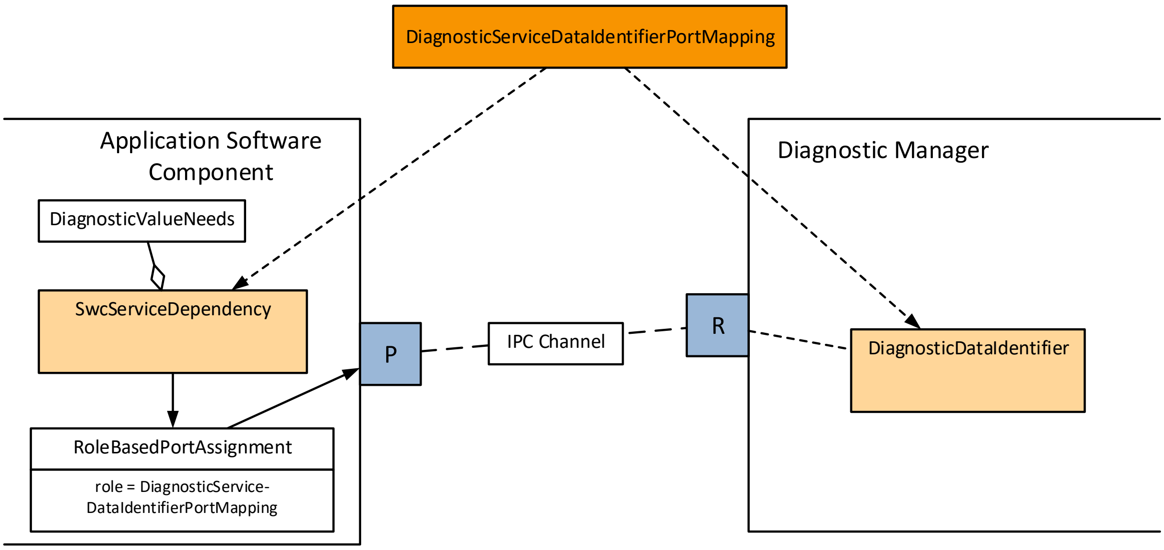


Figure 4.2: Example mapping to associate a `PortPrototype` with a `DiagnosticEvent`

Please note that the mapping targets[1] within a set of diagnostic mappings may exist in several instances at run-time.

This kind of multiple instantiation is formalized by the existence of meta-class `Process` (which in turn is represented by meta-class `ProcessDesign` on design level), see chapter 3.16.

It is very typical that different instances of a piece of application software instances could require a different diagnostic mapping and the modeling needs to accommodate to this requirement, i.e. a relation between a diagnostic mapping and the `ProcessDesign` needs to be established.

As depicted by Figure 4.3, the application of a `DiagnosticMapping` that targets `SwcServiceDependency` on the *AUTOSAR adaptive platform* requires an aggregation chain from of `AdaptiveApplicationSwComponentType` (see section 3.2) via `AdaptiveSwcInternalBehavior` down to `SwcServiceDependency`.

[1] on the end of the application software

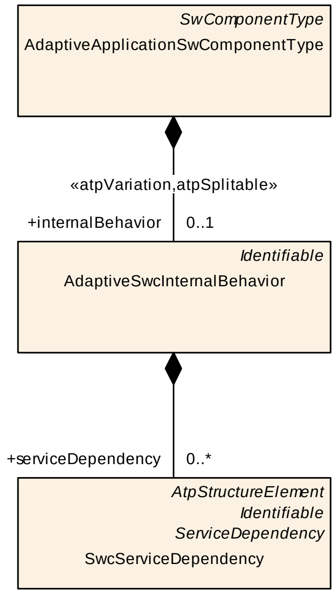


Figure 4.3: Modeling of internal behavior for the modeling of `DiagnosticMapping` that targets `SwcServiceDependency`

Class	**SwcServiceDependency**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
Note	Specialization of ServiceDependency in the context of an SwcInternalBehavior. It allows to associate ports, port groups and (in special cases) data defined for an atomic software component to a given ServiceNeeds element.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceDependency*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
assignedData	RoleBasedData Assignment	*	aggr	Defines the role of an associated data object of the same component. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
assignedPort	RoleBasedPort Assignment	*	aggr	Defines the role of an associated port of the same component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=assignedPort, variationPoint.short Label vh.latestBindingTime=preCompileTime
representedPort Group	PortGroup	0..1	ref	This reference specifies an association between the ServiceNeeeds and a PortGroup, for example to request a communication mode which applies for communication via these ports. The referred PortGroup shall be local to this atomic SWC, but via the links between the Port Groups, a tool can evaluate this information such that all the ports linked via this port group on the same ECU can be found.
serviceNeeds	ServiceNeeds	1	aggr	The associated ServiceNeeds.

Table 4.2: SwcServiceDependency

4.1.2 Diagnostic Event to Port Mapping

[TPS_MANI_01048]{DRAFT} **Mapping of `DiagnosticEvent` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEvent` and one or many `PortPrototype`s is created by using the `DiagnosticEventPortMapping` that refers to a `DiagnosticEvent` in the role `diagnosticEvent` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋*(RS_MANI_00005, RS_MANI_00061)*

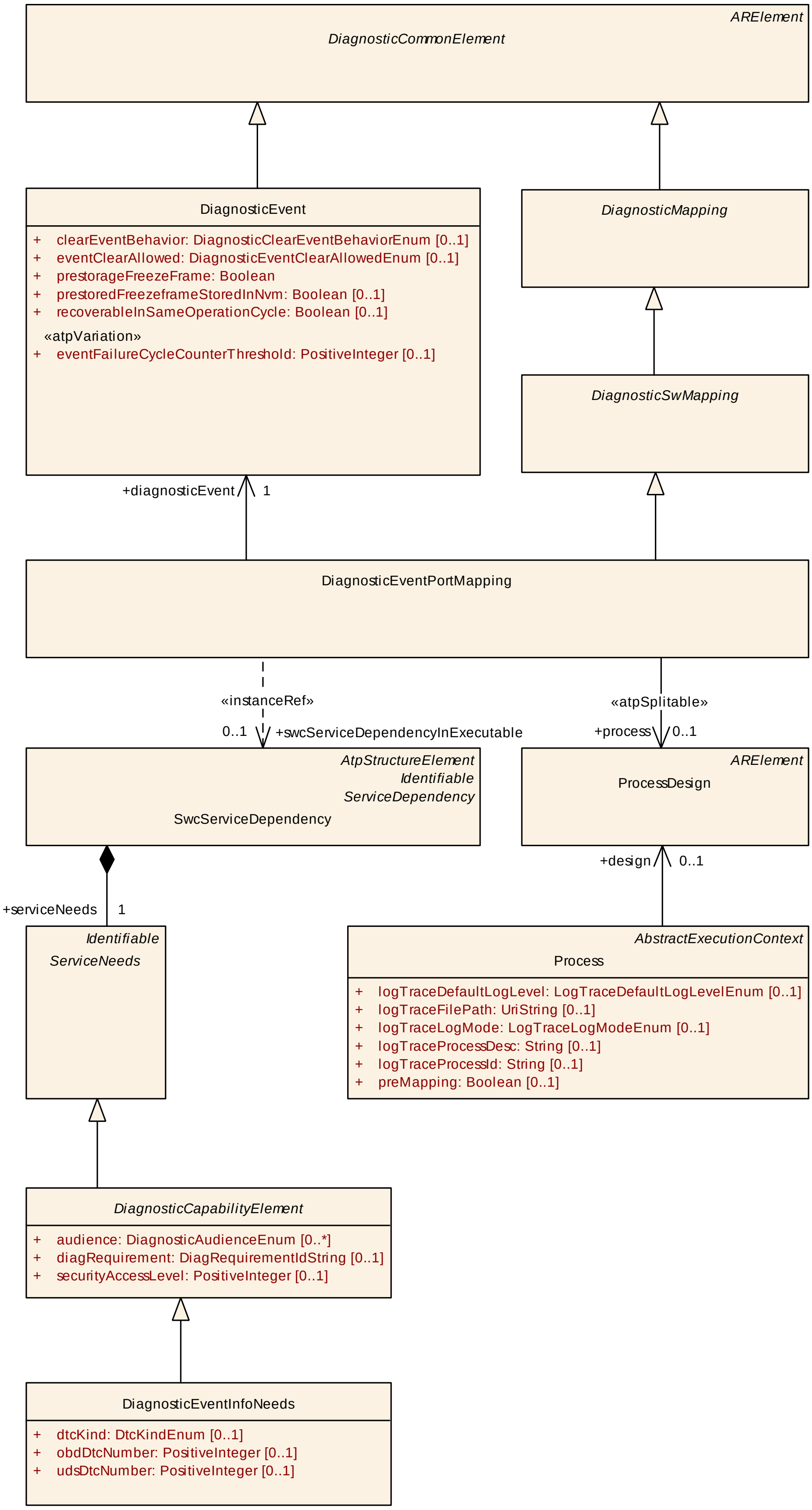


Figure 4.4: Modeling of `DiagnosticEventPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1500]{DRAFT} **Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable** ⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticEventPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior
- aggregate a DiagnosticEventNeeds.

⌋*()*

Class	**DiagnosticEvent**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	This element is used to configure DiagnosticEvents. **Tags:** atp.recommendedPackage=DiagnosticEvents			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
clearEventBehavior	DiagnosticClearEventBehaviorEnum	0..1	attr	This attribute defines the resulting UDS DTC status byte for the related event, which shall not be cleared according to the ClearEventAllowed callback.
connectedIndicator	DiagnosticConnectedIndicator	*	aggr	Event specific description of Indicators. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
eventClearAllowed	DiagnosticEventClearAllowedEnum	0..1	attr	This attribute defines whether the Dem has access to a "ClearEventAllowed" callback.
eventFailureCycleCounterThreshold	PositiveInteger	0..1	attr	This attribute defines the number of failure cycles for the event based fault confirmation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
prestorageFreezeFrame	Boolean	1	attr	This attribute describes whether the Prestorage of Freeze Frames is supported by the assigned event or not. True: Prestorage of FreezeFrames is supported False: Prestorage of FreezeFrames is not supported
prestoredFreezeframeStoredInNvm	Boolean	0..1	attr	If the Event uses a prestored freeze-frame (using the operations PrestoreFreezeFrame and ClearPrestoredFreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm)
recoverableInSameOperationCycle	Boolean	0..1	attr	If the attribute is set to true then reporting PASSED will reset the indication of a failed test in the current operation cycle. If the attribute is set to false then reporting PASSED will be ignored and not lead to a reset of the indication of a failed test.

Table 4.3: DiagnosticEvent

Class	DiagnosticEventPortMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticEventNeeds the DiagnosticEvent is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
bswServiceDependency	BswServiceDependencyIdent	0..1	ref	Reference to a BswServiceDependency that links ServiceNeeds to BswModuleEntries.
diagnosticEvent	DiagnosticEvent	1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports with DiagnosticEventNeeds.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcFlatServiceDependency	SwcServiceDependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This aggregation allows for the usage of the DiagnosticEventPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft
swcServiceDependencyInSystem	SwcServiceDependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.4: DiagnosticEventPortMapping

Class	DiagnosticEventNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Diagnostic Event Manager for one diagnostic event. Its shortName can be regarded as a symbol identifying the diagnostic event from the viewpoint of the component or module which owns this element. In case the diagnostic event specifies a production error, the shortName shall be the name of the production error.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
considerPtoStatus	Boolean	0..1	attr	PTO (Power Take Off) has an impact on the respective emission-related event (OBD). This information shall be provided by SW-C description in order to consider the PTO relevance e.g. for readiness (PID $01) computation. For events with dtcKind set to 'nonEmmissionRelatedDtc' this attribute is typically false.
deferringFid	FunctionInhibitionNeeds	*	ref	This reference contains the link to a function identifier within the FiM which is used by the monitor before delivering a result.
diagEventDebounceAlgorithm	DiagEventDebounceAlgorithm	0..1	aggr	Specifies the abstract need on the Debounce Algorithm applied by the Diagnostic Event Manager.

▽

△

Class	**DiagnosticEventNeeds**			
dtcKind	DtcKindEnum	0..1	attr	This attribute indicates the kind of the diagnostic monitor according to the SWS Diagnostic Event Manger. This attribute applies for the UDS diagnostics use case.
inhibitingFid	FunctionInhibitionNeeds	0..1	ref	This represents the primary Function Inhibition Identifier used for inhibition of the diagnostic monitor. The FID might either inhibit the monitoring of a symptom or the reporting of detected faults.
inhibiting SecondaryFid	FunctionInhibitionNeeds	*	ref	This represents the secondary Function Inhibition Identifier used for inhibition of the diagnostic monitor. Any of the FID inhibitions leads to an inhibition of the monitoring of a symptom or the reporting of detected faults.
obdDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the a function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the OBD diagnostics use case.
prestored Freezeframe StoredInNvm	Boolean	0..1	attr	If the Event uses a prestored freeze-frame (using the operations PrestoreFreezeFrame and ClearPrestored FreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm).
reportBehavior	ReportBehaviorEnum	0..1	attr	This switch indicates whether or not the BSW module is allowed to report the related Events before Dem_Init().
udsDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the a function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the UDS diagnostics use case.
usesMonitor Data	Boolean	0..1	attr	This attribute defines whether additional monitor data shall be added to the reporting of events.

Table 4.5: DiagnosticEventNeeds

4.1.3 Diagnostic Operation Cycle to Port Mapping

[TPS_MANI_01049]{DRAFT} **Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a DiagnosticOperationCycle and one or many PortPrototypes is created by using the DiagnosticOperationCyclePortMapping that refers to a DiagnosticOperationCycle in the role operationCycle as well as to a SwcServiceDependency in the role swcServiceDependencyInExecutable. ⌋*(RS_MANI_00005, RS_MANI_00061)*

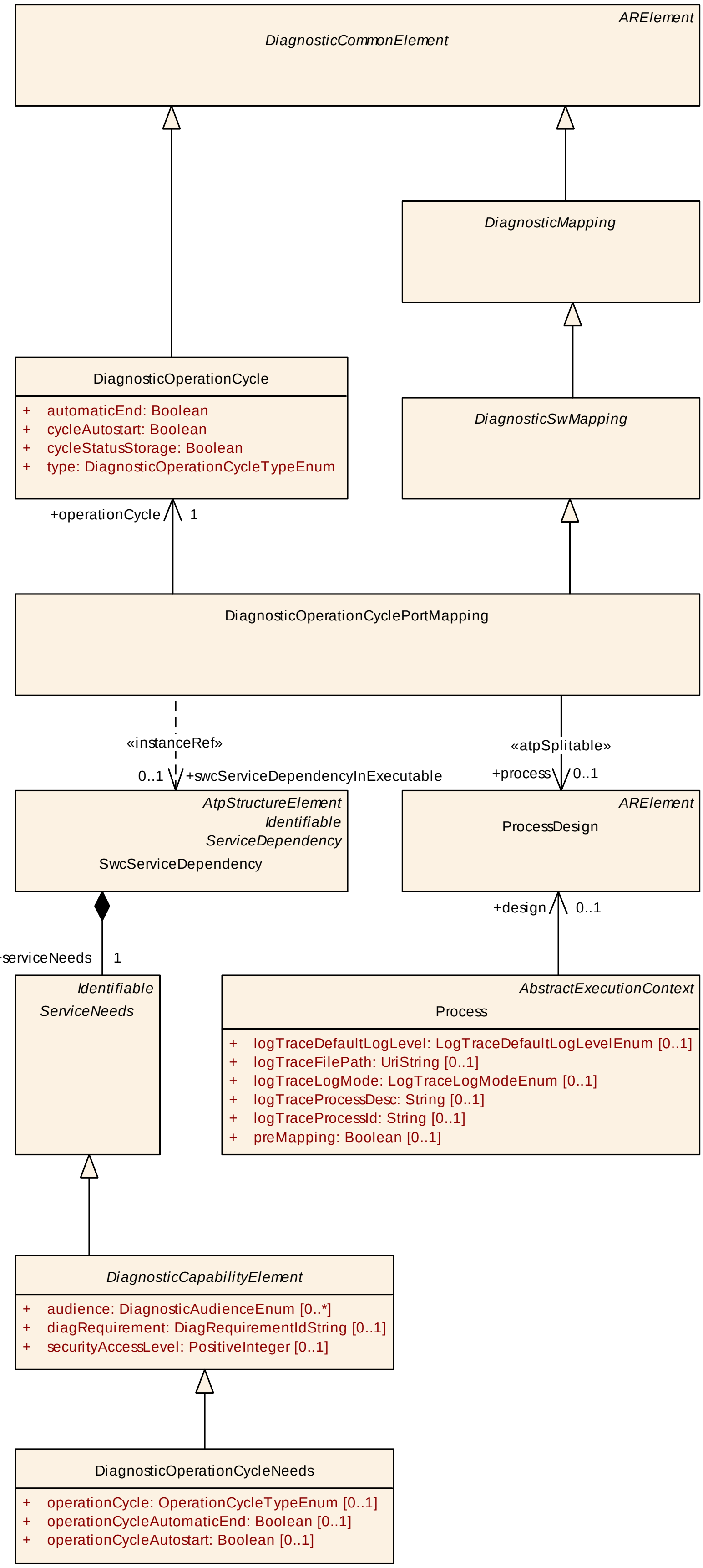


Figure 4.5: Modeling of `DiagnosticOperationCyclePortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1501]{DRAFT} **Target `SwcServiceDependency` of `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable`** ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable` shall

- **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior` and
- aggregate a `DiagnosticOperationCycleNeeds`.

⌋*()*

Class	**DiagnosticOperationCycle**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle			
Note	Definition of an operation cycle that is the base of the event qualifying and for Dem scheduling. **Tags:** atp.recommendedPackage=DiagnosticOperationCycles			
Base	*ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
automaticEnd	Boolean	1	attr	If set to true the driving cycle shall automatically end at either Dem_Shutdown() or Dem_Init().
cycleAutostart	Boolean	1	attr	This attribute defines if the operation cycles is automatically re-started during Dem_PreInit.
cycleStatus Storage	Boolean	1	attr	Defines if the operation cycle state is available over the power cycle (stored non-volatile) or not. • true: the operation cycle state is stored non-volatile • false: the operation cycle state is only stored volatile
type	DiagnosticOperation CycleTypeEnum	1	attr	Operation cycles types for the Dem.

Table 4.6: DiagnosticOperationCycle

Class	**DiagnosticOperationCyclePortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticOperationCycleNeeds the DiagnosticOperationCycle is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
operationCycle	DiagnosticOperation Cycle	1	ref	Reference to the DiagnosticOperationCycle that is assigned to SWC service ports with DiagnosticOperation CycleNeeds.

▽

△

Class	**DiagnosticOperationCyclePortMapping**			
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcFlatService Dependency	SwcService Dependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic OperationCyclePortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft
swcService DependencyIn System	SwcService Dependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.7: DiagnosticOperationCyclePortMapping

Class	**DiagnosticOperationCycleNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide information regarding the operation cycle management to the Dem module.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
operationCycle	OperationCycleType Enum	0..1	attr	Operation cycles types for the Dem to be supported by cycle-state APIs.
operationCycle AutomaticEnd	Boolean	0..1	attr	If this attribute is set to true the Dem shall automatically end the driving cycle at either Dem_Shutdown() or Dem_Init().
operationCycle Autostart	Boolean	0..1	attr	If this attribute is set to true the operation cycles is automatically (re-)started during Dem_PreInit().

Table 4.8: DiagnosticOperationCycleNeeds

4.1.4 Diagnostic Enable Condition to Port Mapping

[TPS_MANI_01050]{DRAFT} **Mapping of `DiagnosticEnableCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEnableCondition` and one or many `PortPrototype`s is created by using the `DiagnosticEnableConditionPortMapping` that refers to a `DiagnosticEnableCondition` in the role `enableCondition` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋*(RS_MANI_00005, RS_MANI_00061)*

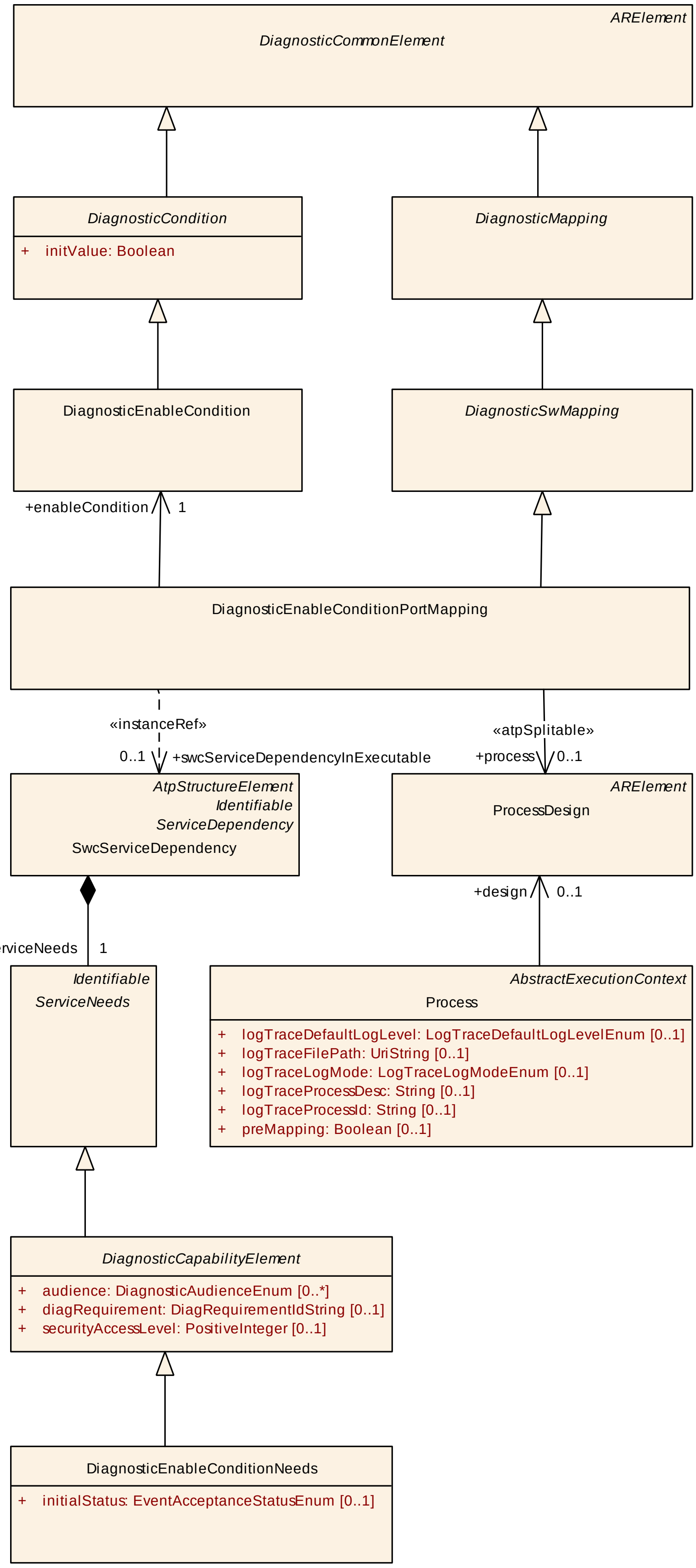


Figure 4.6: Modeling of `DiagnosticEnableConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1502]{DRAFT} **Target `SwcServiceDependency` of `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable`** ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable` shall

- **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior`
- aggregate a `DiagnosticEnableConditionNeeds`.

⌋*()*

Class	**DiagnosticEnableCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of an enable condition. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 4.9: DiagnosticEnableCondition

Class	**DiagnosticEnableConditionPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticEnableConditionNeeds the DiagnosticEnable Condition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
enableCondition	DiagnosticEnable Condition	1	ref	Reference to the EnableCondition which is mapped to a SWC service port with DiagnosticEnableConditionNeeds.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcFlatService Dependency	SwcService Dependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports. This reference can be used in early stages of the development in order to identify the SwcServiceDependency without a full System Context.
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic EnableConditionPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft

▽

△

Class	DiagnosticEnableConditionPortMapping			
swcServiceDependencyInSystem	SwcServiceDependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.10: DiagnosticEnableConditionPortMapping

Class	DiagnosticEnableConditionNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to set an enable condition.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
initialStatus	EventAcceptanceStatusEnum	0..1	attr	Defines the initial status for enable or disable of acceptance of event reports of a diagnostic event.

Table 4.11: DiagnosticEnableConditionNeeds

4.1.5 Diagnostic Storage Condition to Port Mapping

[TPS_MANI_01051]{DRAFT} **Mapping of `DiagnosticStorageCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticStorageCondition` and one or many `PortPrototype`s is created by using the `DiagnosticStorageConditionPortMapping` that refers to a `DiagnosticStorageCondition` in the role `diagnosticStorageCondition` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋*(RS_MANI_00005, RS_MANI_00061)*

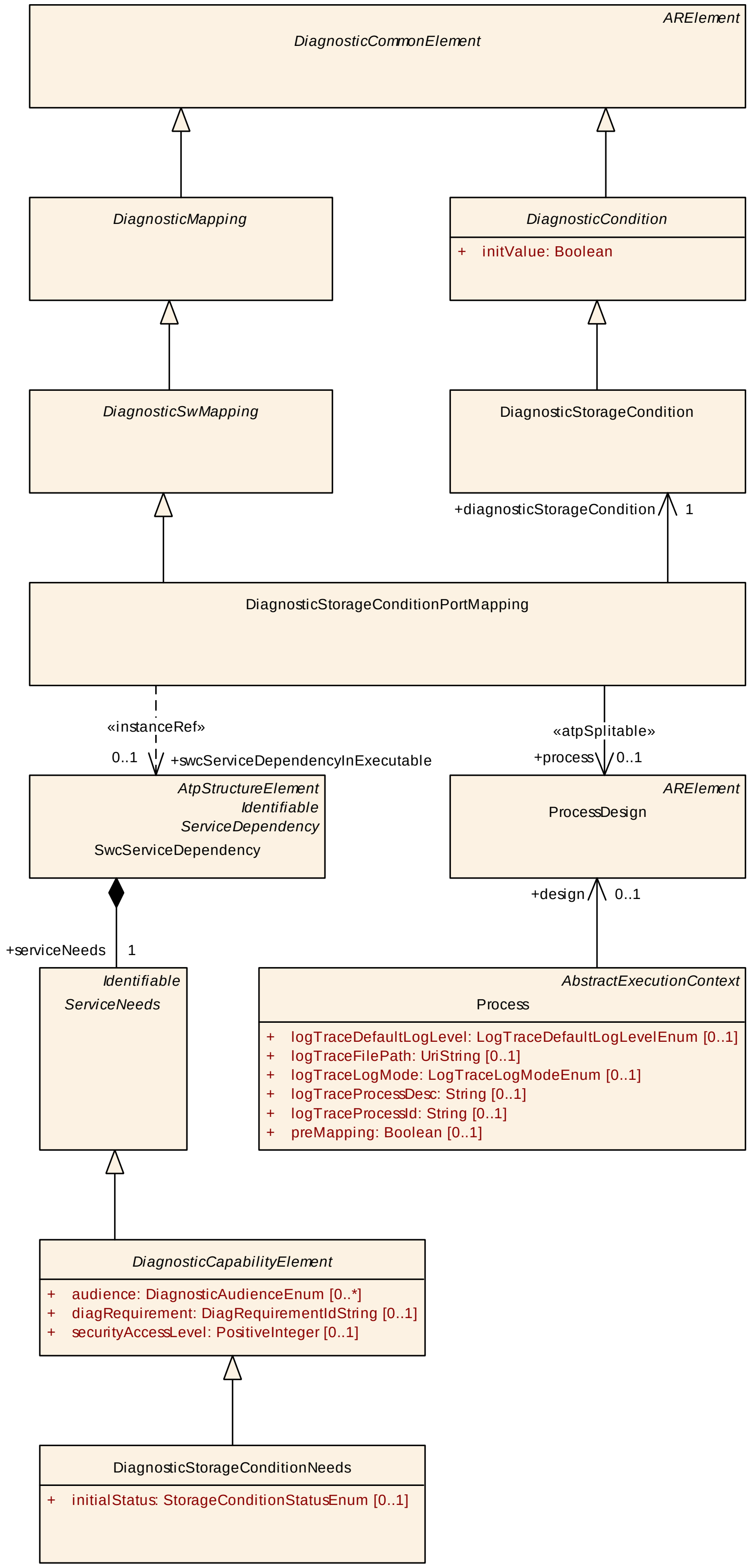


Figure 4.7: Modeling of `DiagnosticStorageConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1503]{DRAFT} **Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable** ⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticStorageConditionNeeds.

⌋*()*

Class	**DiagnosticStorageCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of a storage condition. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 4.12: DiagnosticStorageCondition

Class	**DiagnosticStorageConditionPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticStorageConditionNeeds the DiagnosticStorage Condition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
diagnostic Storage Condition	DiagnosticStorage Condition	1	ref	Reference to the StorageCondition which is mapped to a SWC service port with DiagnosticStorageCondition Needs.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcFlatService Dependency	SwcService Dependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic StorageConditionPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft
swcService DependencyIn System	SwcService Dependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.13: DiagnosticStorageConditionPortMapping

Class	DiagnosticStorageConditionNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to set a storage condition.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
initialStatus	StorageConditionStatus Enum	0..1	attr	Defines the initial status for enable or disable of storage of a diagnostic event.

Table 4.14: DiagnosticStorageConditionNeeds

4.1.6 Diagnostic Clear Condition to Port Mapping

[TPS_MANI_01259]{DRAFT} **Mapping of DiagnosticClearCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a DiagnosticClearCondition and one or many PortPrototypes is created by using the DiagnosticClearConditionPortMapping that refers to a DiagnosticClearCondition in the role clearCondition as well as to a SwcServiceDependency in the role swcServiceDependencyInExecutable. ⌋*(RS_MANI_00005, RS_MANI_00061)*

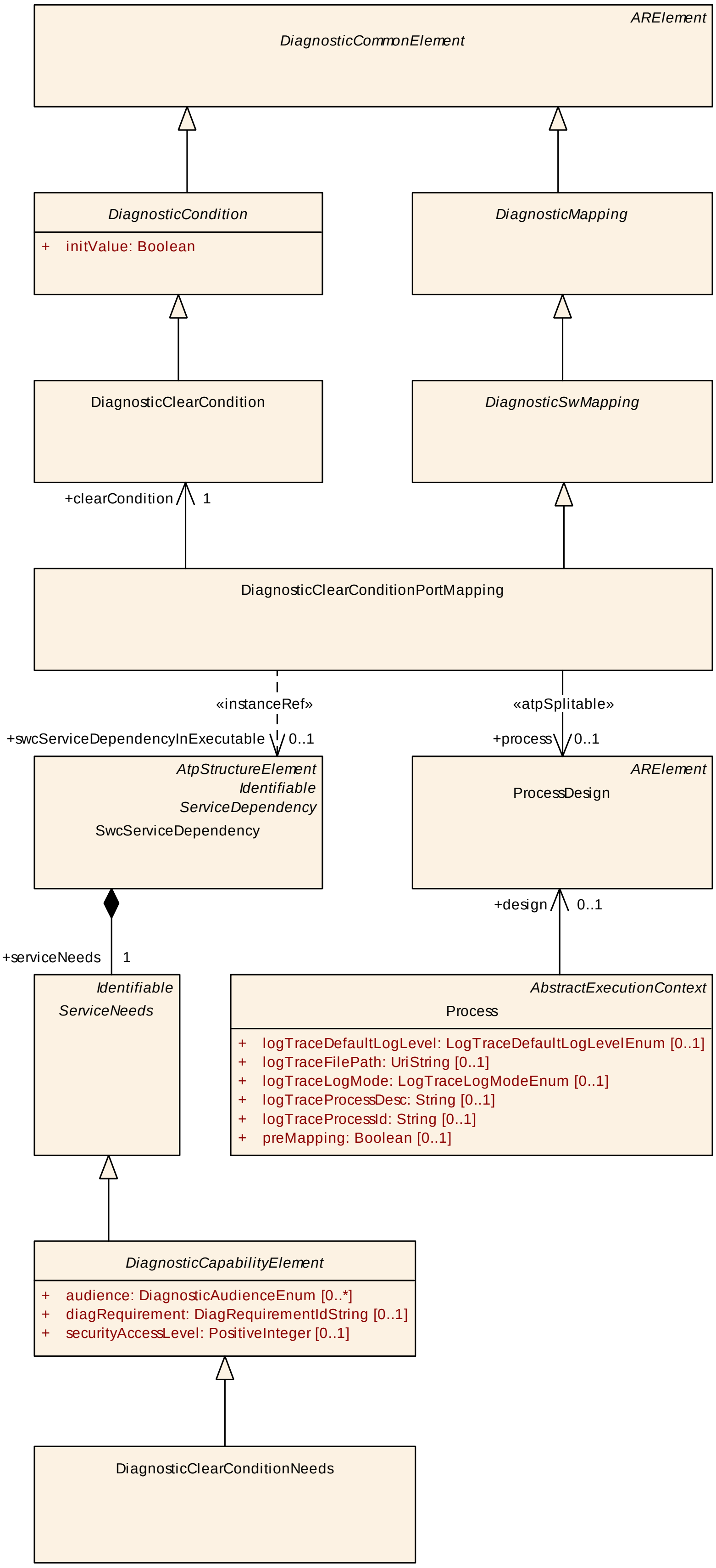


Figure 4.8: Modeling of `DiagnosticClearConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1698]{DRAFT} **Target SwcServiceDependency of DiagnosticClearConditionPortMapping.swcServiceDependencyInExecutable** ⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticClearConditionPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticClearConditionNeeds.

⌋*()*

Class	**DiagnosticClearConditionPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsClearConditionNeeds the DiagnosticClearCondition is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
clearCondition	DiagnosticClearCondition	1	ref	Reference to the ClearCondition which is mapped to a SWC service port with DiagnosticClearConditionNeeds. **Tags:** atp.Status=draft
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This aggregation allows for the usage of the DiagnosticClearConditionPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft

Table 4.15: DiagnosticClearConditionPortMapping

4.1.7 Diagnostic Indicator to Port Mapping

[TPS_MANI_01260]{DRAFT} **Mapping of DiagnosticIndicator to PortPrototype(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a DiagnosticIndicator and one or many PortPrototypes is created by using the DiagnosticIndicatorPortMapping that refers to a DiagnosticIndicator in the role indicator as well as to a SwcServiceDependency in the role swcServiceDependencyInExecutable. ⌋*(RS_MANI_00005, RS_MANI_00061)*

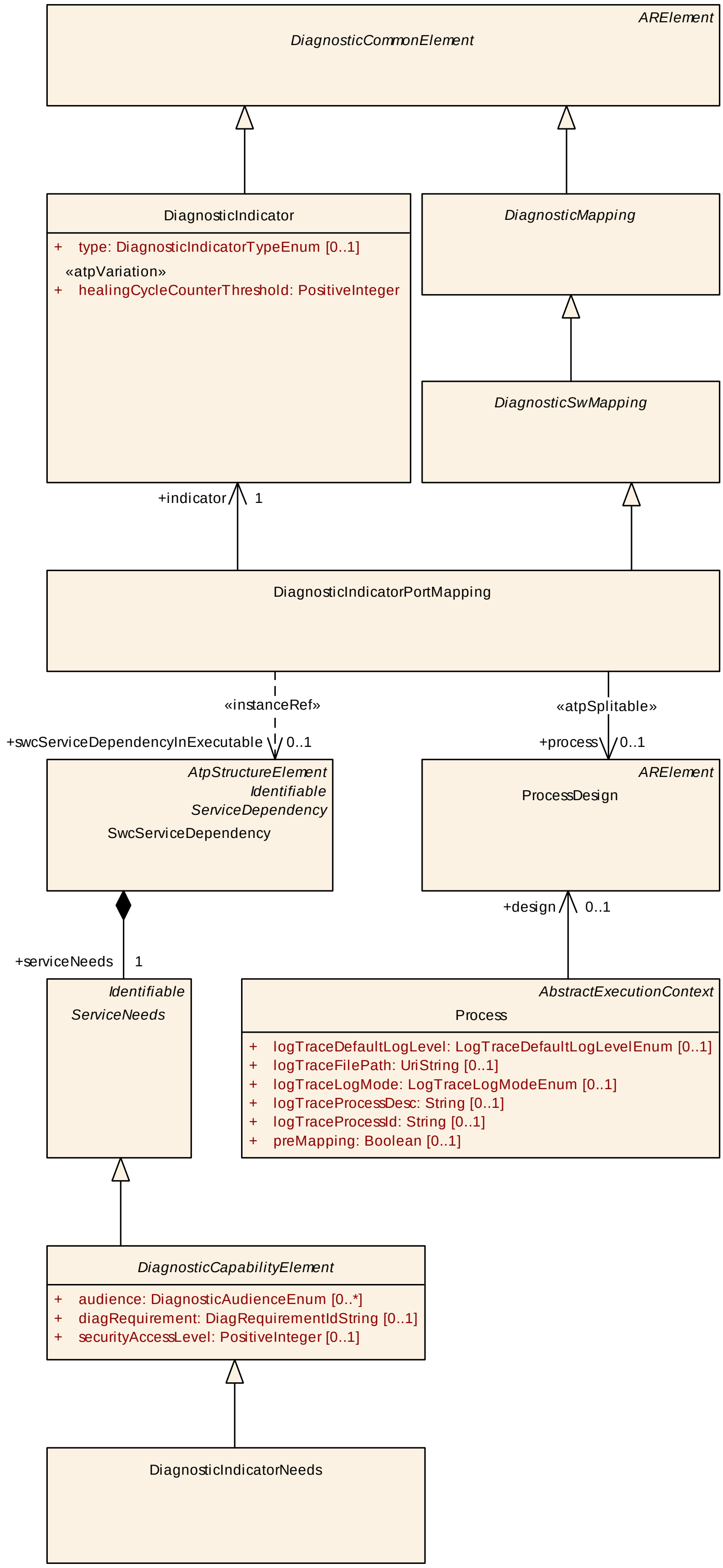


Figure 4.9: Modeling of `DiagnosticIndicatorPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1699]{DRAFT} **Target SwcServiceDependency of DiagnosticIndicatorPortMapping.swcServiceDependencyInExecutable** ⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticIndicatorPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticIndicatorNeeds.

⌋*()*

Class	**DiagnosticIndicatorPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsIndicatorNeeds the DiagnosticIndicator is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
indicator	DiagnosticIndicator	1	ref	Reference to the DiagnosticIndicator which is mapped to a SWC service port with DiagnosticIndicatorNeeds. **Tags:** atp.Status=draft
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic IndicatorPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft

Table 4.16: DiagnosticIndicatorPortMapping

4.1.8 Diagnostic Memory Destination to Port Mapping

[TPS_MANI_01261]{DRAFT} **Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a DiagnosticMemoryDestination and one or many PortPrototypes is created by using the DiagnosticMemoryDestinationPortMapping that refers to a DiagnosticMemoryDestination in the role memoryDestination as well as to a SwcServiceDependency in the role swcServiceDependencyInExecutable. ⌋*(RS_MANI_00005, RS_MANI_00061)*

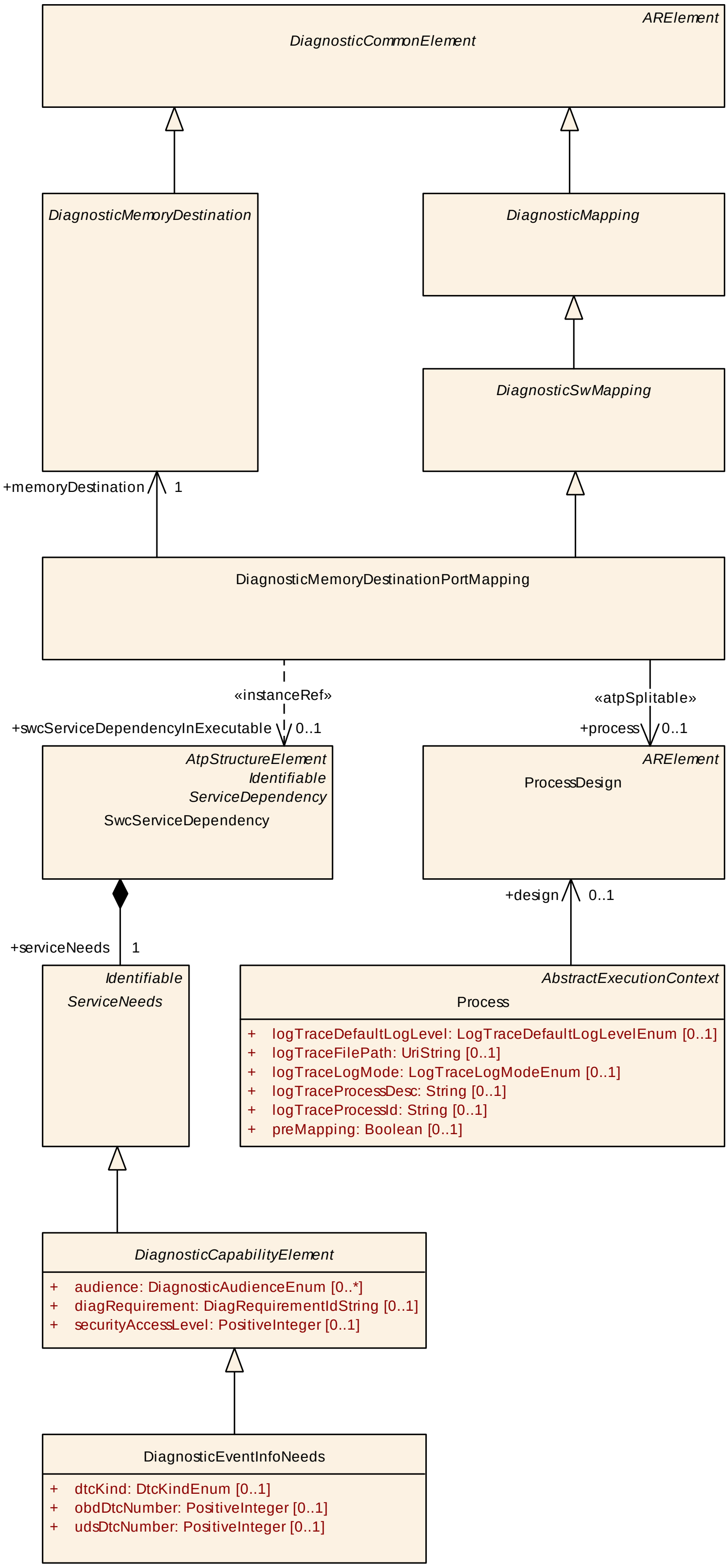


Figure 4.10: Modeling of `DiagnosticMemoryDestinationPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1700]{DRAFT} **Target SwcServiceDependency of DiagnosticMemoryDestinationPortMapping.swcServiceDependencyInExecutable** ⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticMemoryDestinationPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticEventInfoNeeds.

⌋*()*

Class	**DiagnosticMemoryDestinationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsEventInfoNeeds the DiagnosticMemoryDestination is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
memoryDestination	DiagnosticMemoryDestination	1	ref	Reference to the MemoryDestination which is mapped to a SWC service port with DiagnosticEventInfoNeeds. **Tags:** atp.Status=draft
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This aggregation allows for the usage of the DiagnosticMemoryDestinationMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft

Table 4.17: DiagnosticMemoryDestinationPortMapping

Class	**DiagnosticEventInfoNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component interested to get information regarding specific DTCs.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dtcKind	DtcKindEnum	0..1	attr	This attribute indicates the kind of the diagnostic event according to the SWS Diagnostic Event Manger for which the DiagnosticInfo is requested. This attribute applies for the UDS diagnostics use case.

▽

△

Class	**DiagnosticEventInfoNeeds**			
obdDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the OBD diagnostics use case.
udsDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the UDS diagnostics use case.

Table 4.18: DiagnosticEventInfoNeeds

4.1.9 Diagnostic Security to Port Mapping

[TPS_MANI_01262]{DRAFT} **Mapping of `DiagnosticSecurityLevel` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSecurityLevel` and one or many `PortPrototype`s is created by using the `DiagnosticSecurityLevelPortMapping` that refers to a `DiagnosticSecurityLevel` in the role `securityLevel` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋*(RS_MANI_00005, RS_MANI_00061)*

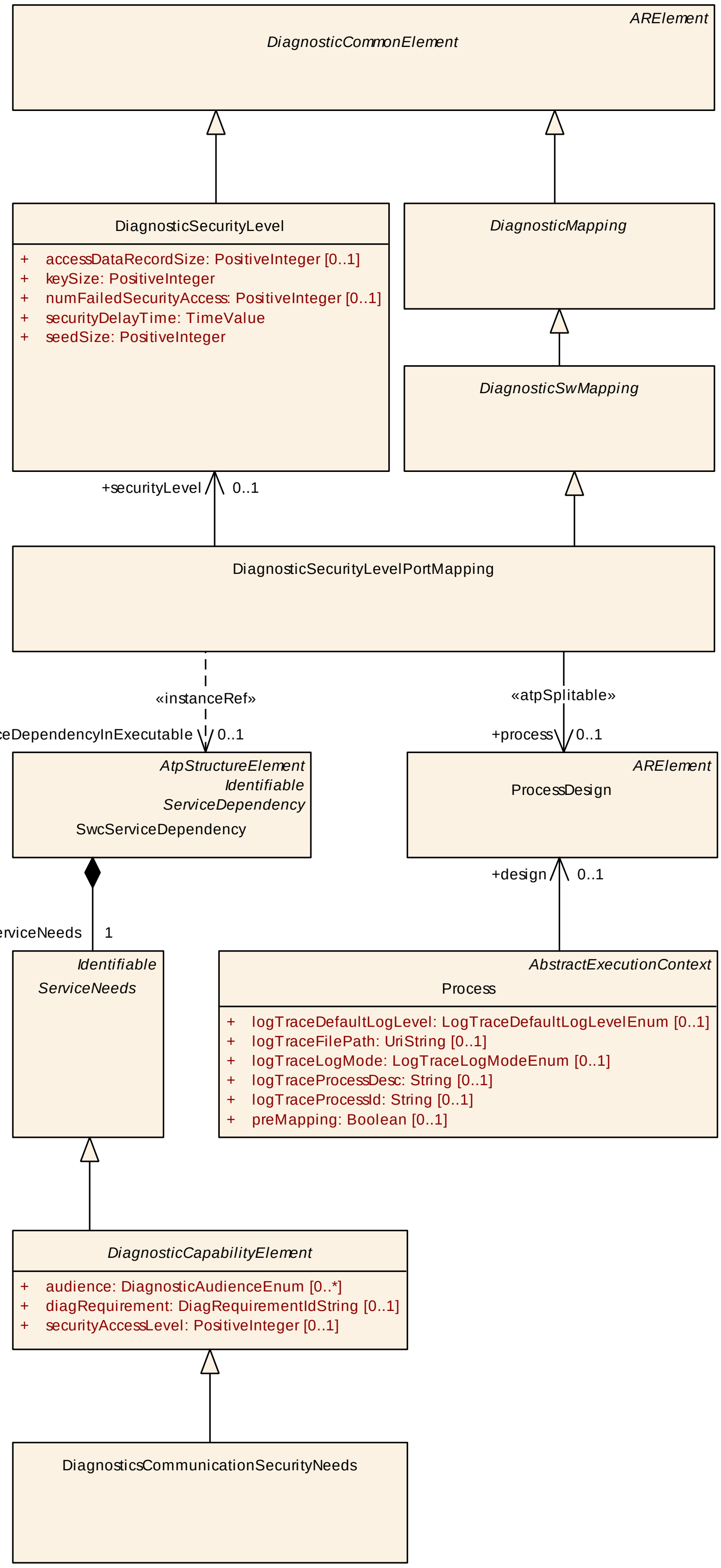


Figure 4.11: Modeling of `DiagnosticSecurityLevelPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1701]{DRAFT} **Target** **SwcServiceDependency** **of** **DiagnosticSecurityLevelPortMapping.swcServiceDependencyInExecutable** ⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticSecurityLevelPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticsCommunicationSecurityNeeds.

⌋*()*

Class	**DiagnosticSecurityLevelPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsCommunicationSecurityNeeds the Diagnostic SecurityLevel is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
securityLevel	DiagnosticSecurityLevel	0..1	ref	Reference to the SecurityLevelwhich is mapped to a SWC service port with DiagnosticCommunicationSecurity Needs. **Tags:** atp.Status=draft
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the Diagnostic SecurityLevelMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft

Table 4.19: DiagnosticSecurityLevelPortMapping

Class	**DiagnosticsCommunicationSecurityNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to verify the access to security level via diagnostic services.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 4.20: DiagnosticsCommunicationSecurityNeeds

4.1.10 Diagnostic Data Identifier to Port Mapping

The DM on the *AUTOSAR adaptive platform* has the ability to access entire DiagnosticDataIdentifiers at once. For supporting this ability, a dedicated mapping class named DiagnosticServiceDataIdentifierPortMapping is introduced.

[TPS_MANI_01263]{DRAFT} **Mapping of DiagnosticDataIdentifier to PortPrototype(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between a DiagnosticDataIdentifier and one or many PortPrototypes is created by using the DiagnosticServiceDataIdentifierPortMapping that refers to a DiagnosticDataIdentifier in the role diagnosticDataIdentifier as well as to a SwcServiceDependency in the role swcServiceDependencyInExecutable. ⌋*(RS_MANI_00005, RS_MANI_00061)*

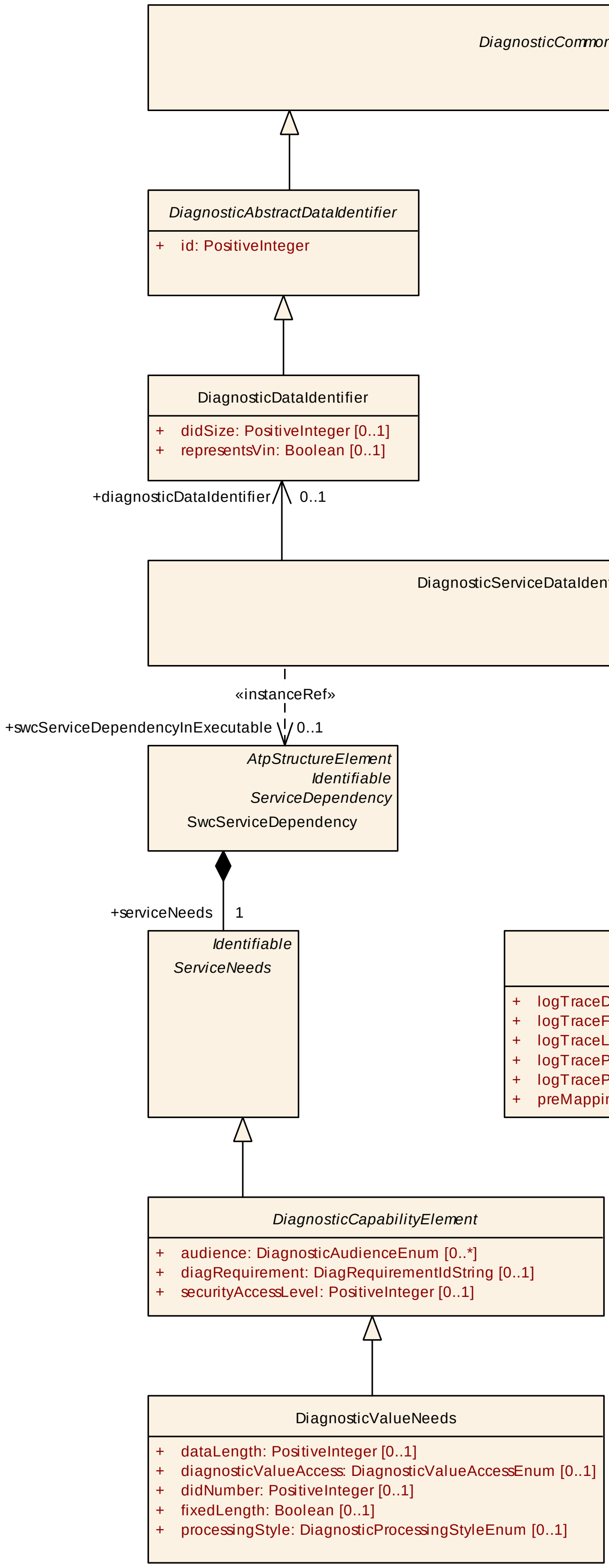


Figure 4.12: Modeling of `DiagnosticServiceDataIdentifierPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1702]{DRAFT} **Target SwcServiceDependency of DiagnosticServiceDataIdentifierPortMapping.swcServiceDependencyInExecutable**
⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticServiceDataIdentifierPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticValueNeeds.

⌋*()*

Class	DiagnosticServiceDataIdentifierPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class provides the ability to define a diagnostic access to an entire DID. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
diagnosticDataIdentifier	DiagnosticDataIdentifier	0..1	ref	This reference represents the applicable DiagnosticDataIdentfiier. **Tags:** atp.Status=draft
process	ProcessDesign	1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This reference identifies the applicable SwcServiceDependency. The reference has the ability to point into the component hierarchy (under possible consideration of the rootSoftwareComposition). **Tags:** atp.Status=draft

Table 4.21: DiagnosticServiceDataIdentifierPortMapping

Class	DiagnosticValueNeeds
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	Specifies the general needs on the configuration of the Diagnostic Communication Manager (DCM) which are not related to a particular item (e.g. a PID). The main use case is the mapping of service ports to the DCM which are not related to a particular item. In the case of using a sender receiver communicated value, the related value shall be taken via assigned Data in the role "signalBasedDiagnostics". In case of using a client/server communicated value, the related value shall be communicated via the port referenced by asssignedPort. The details of this communication (e.g. appropriate naming conventions) are specified in the related software specifications (SWS).
Base	*ARObject, DiagnosticCapabilityElement, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds*

▽

△

Class	DiagnosticValueNeeds			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataLength	PositiveInteger	0..1	attr	This attribute is applicable only if the ServiceNeed is aggregated within BswModuleDependency. This attribute represents the length of data (in bytes) provided for this particular PID signal.
diagnosticValueAccess	DiagnosticValueAccessEnum	0..1	attr	This attribute controls whether the data can be read and written or whether it is to be handled read-only.
didNumber	PositiveInteger	0..1	attr	This represents a Data identifier for the diagnostic value. This allows to predefine the DID number if the responsible function developer has received a particular requirement from the OEM or from a standardization body.
fixedLength	Boolean	0..1	attr	This attribute controls whether the data length of the data is fixed.
processingStyle	DiagnosticProcessingStyleEnum	0..1	attr	This attribute controls whether interaction requires the software-component to react synchronously on a request or whether it processes the request in background but still the DCM has to issue the call again to eventually obtain the result of the request.

Table 4.22: DiagnosticValueNeeds

4.1.11 Diagnostic Generic UDS Service Handler to Port Mapping

[TPS_MANI_01264]{DRAFT} **Mapping of `DiagnosticServiceInstance` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between **a collection of** `DiagnosticServiceInstance`s and one or many `PortPrototype`s is created by using the `DiagnosticGenericUdsPortMapping` that refers to a `DiagnosticServiceInstance` in the role `serviceInstance` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋*(RS_MANI_00005, RS_MANI_00061)*

In other words, it is possible to associate a collection of UDS services to a given `PPortPrototype` with the intention that the `PPortPrototype` can handle the associated services.

By creating a dedicated association between generic UDs handlers and the services they can take it is possible to use multiple generic UDS handlers and let each take only the associated services.

Technically, a possible alternative to the documented modeling of generic UDS handling would be to avoid the mapping at all and foresee the existence of a catch-all generic UDS handler.

This, to a large extent, contradicts the idea of having modular software installations on the basis of the definition of `SoftwareCluster`s (see section 12.3).

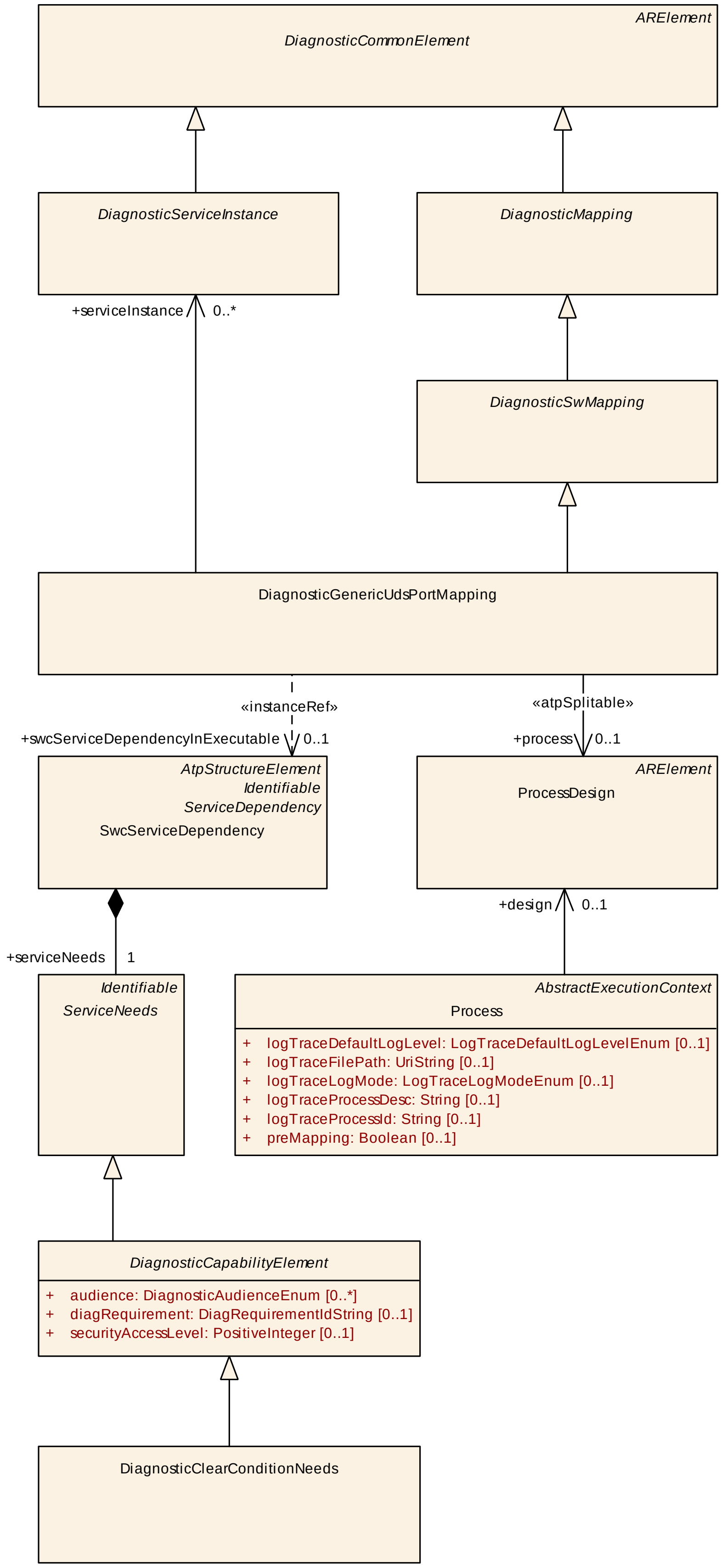


Figure 4.13: Modeling of `DiagnosticGenericUdsPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1703]{DRAFT} **Target SwcServiceDependency of DiagnosticGenericUdsPortMapping.swcServiceDependencyInExecutable** ⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticGenericUdsPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticGenericUdsNeeds.

⌋*()*

Class	**DiagnosticGenericUdsPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticsGenericUdsNeeds a collection of DiagnosticServiceInstances is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
serviceInstance	DiagnosticServiceInstance	*	ref	Reference to the ServiceInstances mapped to a SWC service port with DiagnosticGenericUdsNeeds. **Tags:** atp.Status=draft
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This aggregation allows for the usage of the DiagnosticGenericUdsPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft

Table 4.23: DiagnosticGenericUdsPortMapping

4.1.12 Diagnostic Upload/Download Port Mapping

[TPS_MANI_01266]{DRAFT} **Mapping of DiagnosticServiceInstance for upload/download to PortPrototype(s) on the *AUTOSAR adaptive platform*** ⌈ On the *AUTOSAR adaptive platform*, the relation between **a collection of** DiagnosticServiceInstances and one or many PortPrototypes for upload/download purposes is created by using the DiagnosticUploadDownloadPortMapping that refers to a DiagnosticServiceInstance in the role serviceInstance as well as to a SwcServiceDependency in the role swcServiceDependencyInExecutable. ⌋ *(RS_MANI_00005, RS_MANI_00061)*

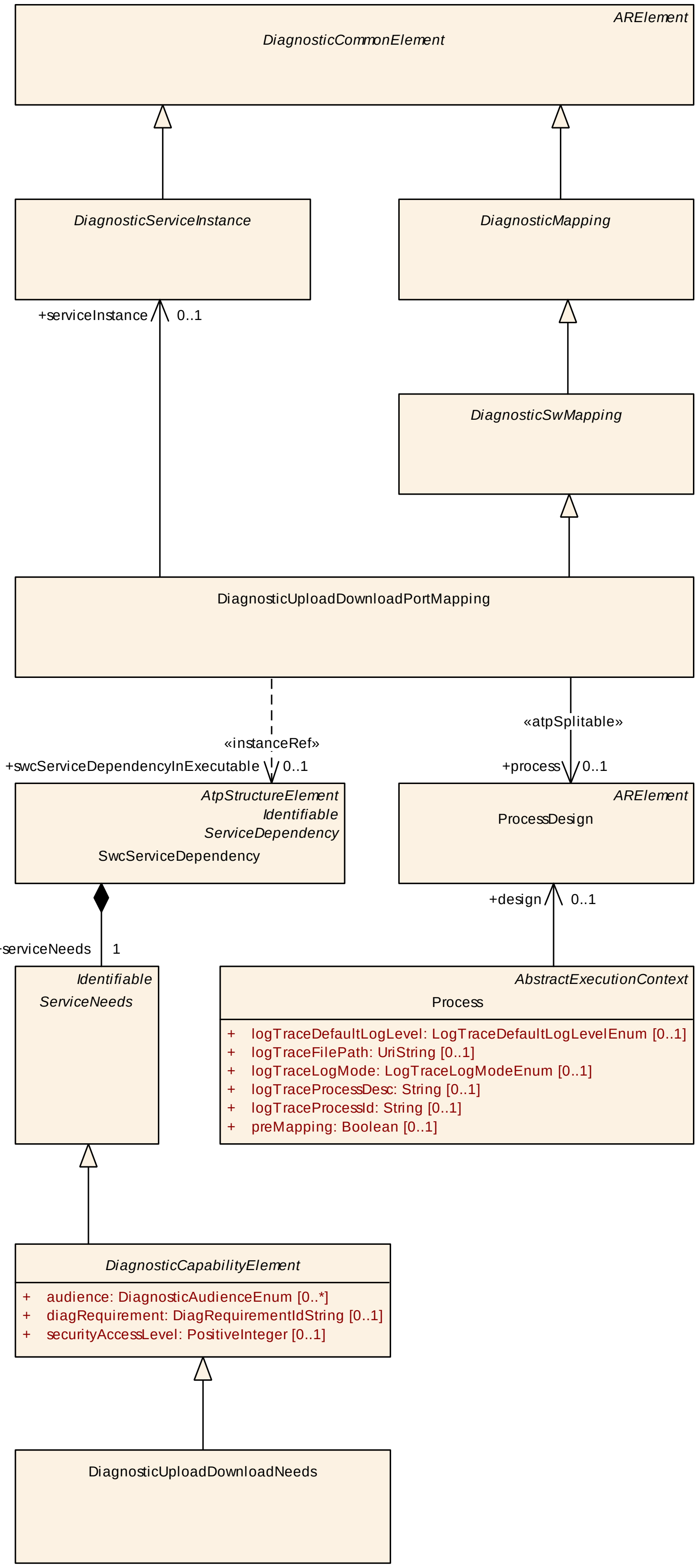


Figure 4.14: Modeling of `DiagnosticUploadDownloadPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1704]{DRAFT} **Target SwcServiceDependency of DiagnosticUploadDownloadPortMapping.swcServiceDependencyInExecutable** ⌈ Any particular SwcServiceDependency that is referenced in the role DiagnosticUploadDownloadPortMapping.swcServiceDependencyInExecutable shall

- **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior and
- aggregate a DiagnosticUploadDownloadNeeds.

⌋*()*

Class	**DiagnosticUploadDownloadPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticUploadDownloadNeeds the DiagnosticServiceInstance is mapped. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
serviceInstance	DiagnosticService Instance	0..1	ref	Reference to the ServiceInstance mapped to a SWC service port with DiagnosticUploadDownloadNeeds. **Tags:** atp.Status=draft
swcService DependencyIn Executable	SwcService Dependency	0..1	iref	This aggregation allows for the usage of the DiagnosticUploadDownloadPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft

Table 4.24: DiagnosticUploadDownloadPortMapping

Class	**DiagnosticUploadDownloadNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to specify needs regarding upload and download by means of diagnostic services.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 4.25: DiagnosticUploadDownloadNeeds

4.1.13 Diagnostic Data Mapping

Disclaimer: The DiagnosticServiceDataMapping is currently not supported as input for the configuration of AUTOSAR Adaptive Diagnostic Management.

[TPS_MANI_01037]{DRAFT} **Diagnostic data mapping on the *AUTOSAR adaptive platform*** ⌈ The diagnostic data mapping on the *AUTOSAR adaptive platform* is created by means of meta-class DiagnosticServiceDataMapping that maps a DiagnosticDataElement to a DataPrototype referenced in the role mappedApDataElement. ⌋*(RS_MANI_00005)*

[TPS_MANI_01060]{DRAFT} **Use cases for the application of DiagnosticServiceDataMapping** ⌈ DiagnosticServiceDataMapping shall only be used where access to data is free of side-effects. This is the case for the notifier events of fields and, at least with respect to the value, events. ⌋*(RS_MANI_00005)*

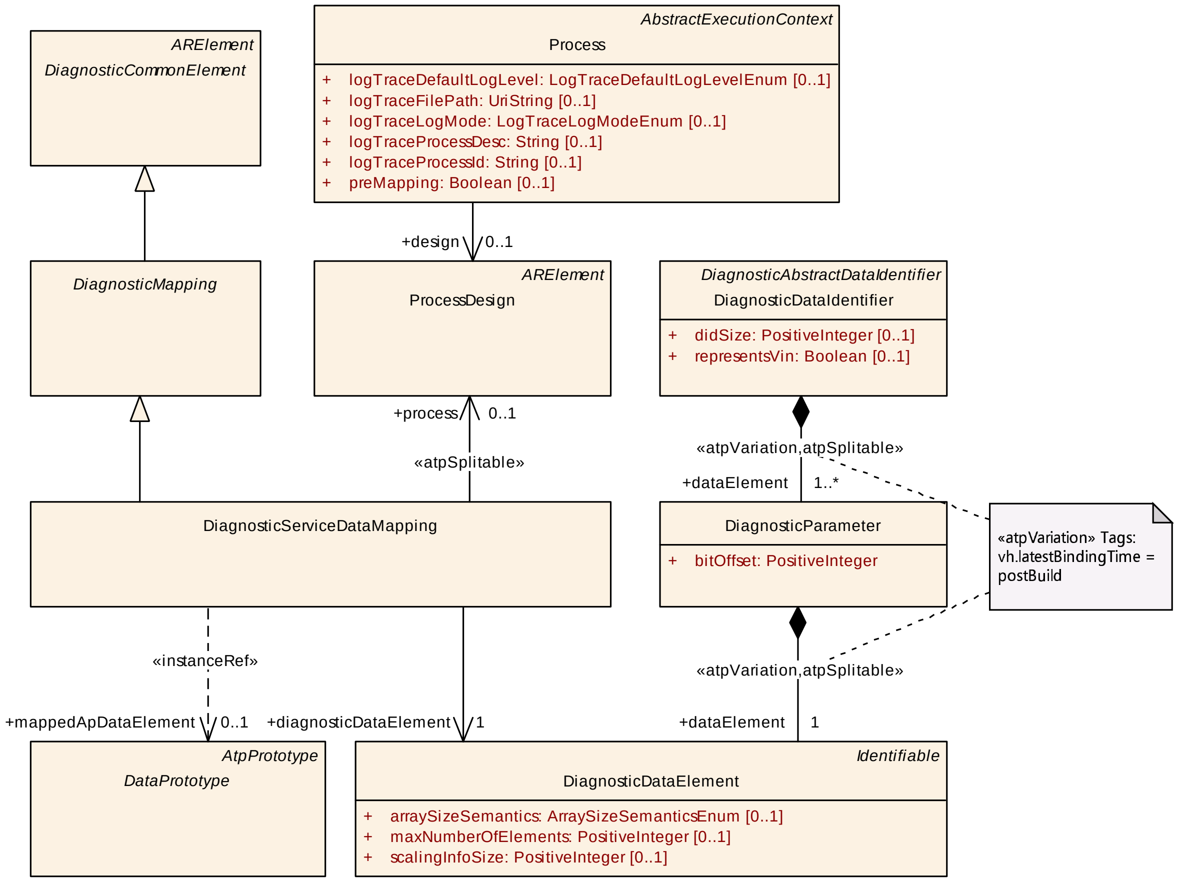


Figure 4.15: Modeling of the diagnostic data mapping

Please note that the DiagnosticServiceDataMapping can be applied on models on the *AUTOSAR adaptive platform* because the mapping target is a DataPrototype that is aggregated by a ServiceInterface in the context of a PortPrototype.

In other words, the DiagnosticServiceDataMapping applies for the mapping to an event or field, **or even to an element of** an event or field.

[constr_1496]{DRAFT} **DiagnosticServiceDataMapping.mappedApDataElement shall only refer to specific sub-classes of DataPrototype** ⌈ A DiagnosticServiceDataMapping.mappedApDataElement shall only refer to an event or a field or a DataPrototype owned by an event or a field. ⌋*()*

Please note that the existence of [constr_1496] is a direct consequence of the existence of [TPS_MANI_01060].

In particular, [constr_1496] prevents the creation of a `DiagnosticServiceDataMapping` to a `ArgumentDataPrototype`. In the diagnostic context, `ArgumentDataPrototype` are mainly used in the argument list of the sub-functions of diagnostic routines which are rarely free of side-effects.

Class	**DiagnosticServiceDataMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::ServiceMapping			
Note	This represents the ability to define a mapping of a diagnostic service to a software-component. This kind of service mapping is applicable for the usage of SenderReceiverInterfaces or event/notifier semantics in ServiceInterfaces on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
diagnosticData Element	DiagnosticDataElement	1	ref	This represents the applicable payload that corresponds to the referenced DataPrototype in the role mappedData Element or (in case of a usage on the adaptive platform) mappedApDataElement.
mappedApData Element	DataPrototype	0..1	iref	This represents the dataElement in the application software of an adaptive AUTOSAR application that is accessed for diagnostic purpose. **Tags:** atp.Status=draft
mappedData Element	DataPrototype	0..1	iref	This represents the dataElement in the application software that is accessed for diagnostic purpose. This role is applicable on the classic platform.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft

Table 4.26: DiagnosticServiceDataMapping

Class	**DiagnosticDataElement**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to describe a concrete piece of data to be taken into account for diagnostic purposes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOf Elements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of how many elements the array can take.
scalingInfoSize	PositiveInteger	0..1	attr	Size in bytes of scaling information for the DiagnosticData Element if used with DiagnosticReadScalingDataBy Identifier

▽

△

Class	**DiagnosticDataElement**			
swDataDef Props	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties in order to support the definition of e.g. computation formulae and data constraints.

Table 4.27: DiagnosticDataElement

4.2 Diagnostic Clear Condition

On the *AUTOSAR adaptive platform*, a new model element similar DiagnosticEnableCondition or DiagnosticStorageCondition is introduced: DiagnosticClearCondition.

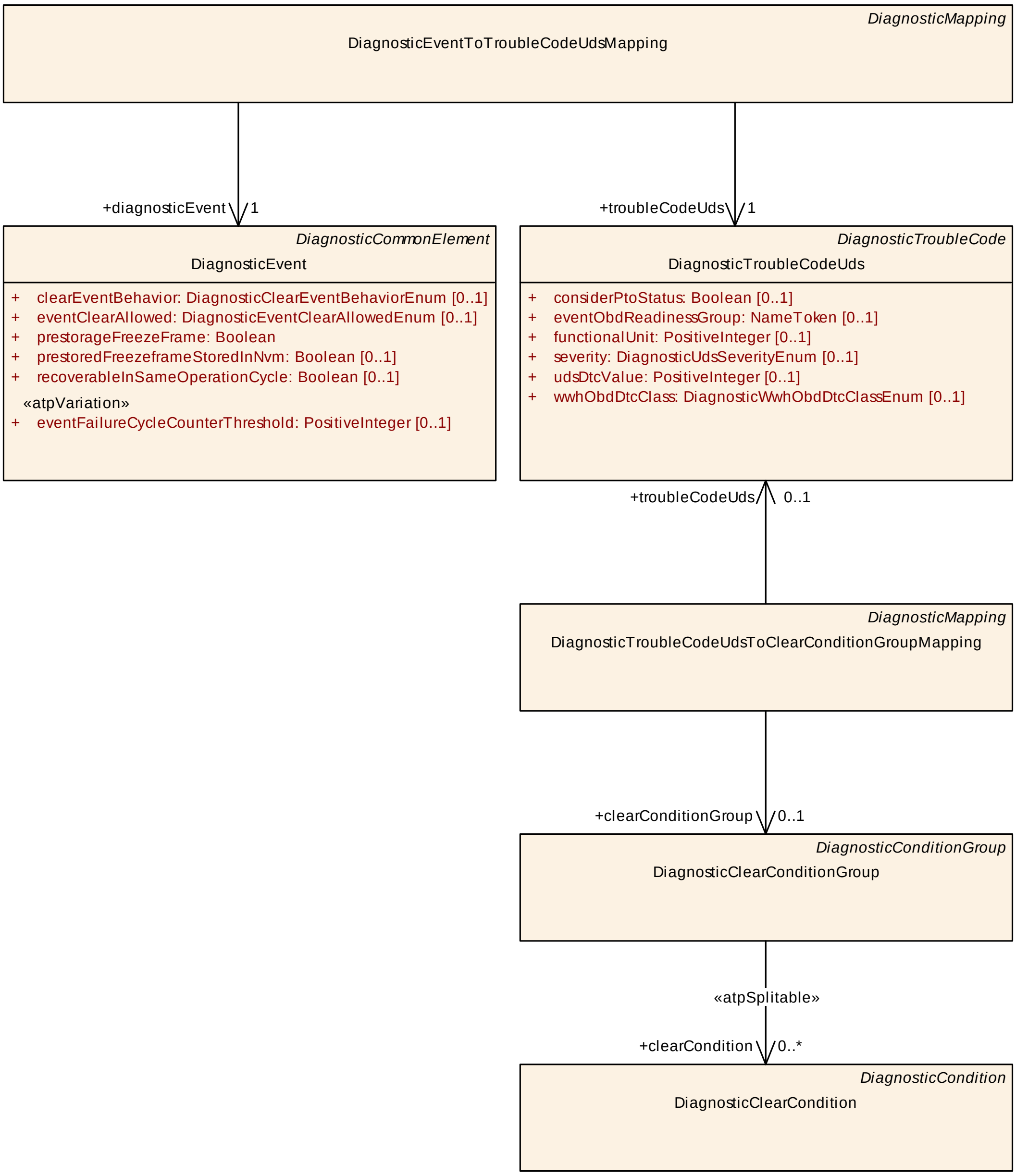


Figure 4.16: Modeling of the diagnostic clear condition

In contrast to DiagnosticEnableCondition or DiagnosticStorageCondition, DiagnosticClearCondition is not mapped to a DiagnosticEvent but (via the aggregation by DiagnosticClearConditionGroup) to a DiagnosticTroubleCodeUds.

For this purpose, meta-class DiagnosticTroubleCodeUdsToClearConditionGroupMapping has been defined.

Class	**DiagnosticClearCondition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	This meta-class describes a clear condition for diagnostic purposes. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 4.28: DiagnosticClearCondition

Class	**DiagnosticClearConditionGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	Clear condition group which includes one or several clear conditions. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticClearConditionGroups			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticConditionGroup*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
clearCondition	DiagnosticClear Condition	*	ref	This aggregation represents the collection of Diagnostic ClearConditions that belong to the DiagnosticClear ConditionGroup. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=clearCondition atp.Status=draft

Table 4.29: DiagnosticClearConditionGroup

Class	**DiagnosticTroubleCodeUdsToClearConditionGroupMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
clearCondition Group	DiagnosticClear ConditionGroup	0..1	ref	Thi reference identifies the applicable DiagnosticClear ConditionGroup. **Tags:** atp.Status=draft

▽

△

Class	DiagnosticTroubleCodeUdsToClearConditionGroupMapping			
troubleCodeUds	DiagnosticTroubleCodeUds	0..1	ref	This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping. **Tags:** atp.Status=draft

Table 4.30: DiagnosticTroubleCodeUdsToClearConditionGroupMapping

[constr_1658]{DRAFT} **Number of DiagnosticTroubleCodeUdsToClearConditionGroupMapping elements per DiagnosticTroubleCodeUds** ⌈ The mapping element DiagnosticTroubleCodeUdsToClearConditionGroupMapping shall be created no more than once per DiagnosticTroubleCodeUds.

If several DiagnosticTroubleCodeUdsToClearConditionGroupMapping elements referring the same DiagnosticTroubleCodeUds are defined, then the Clear Condition Group mapping shall be regarded as defective. ⌋*()*

4.3 Security Access

the implementation of the diagnostics manager on the adaptive platform requires a refined modeling of meta-class DiagnosticSecurityAccessClass.

A new attribute named sharedTimer is introduced that controls whether a single timer is used for all security access levels or whether the individual levels utilize separate timers respectively.

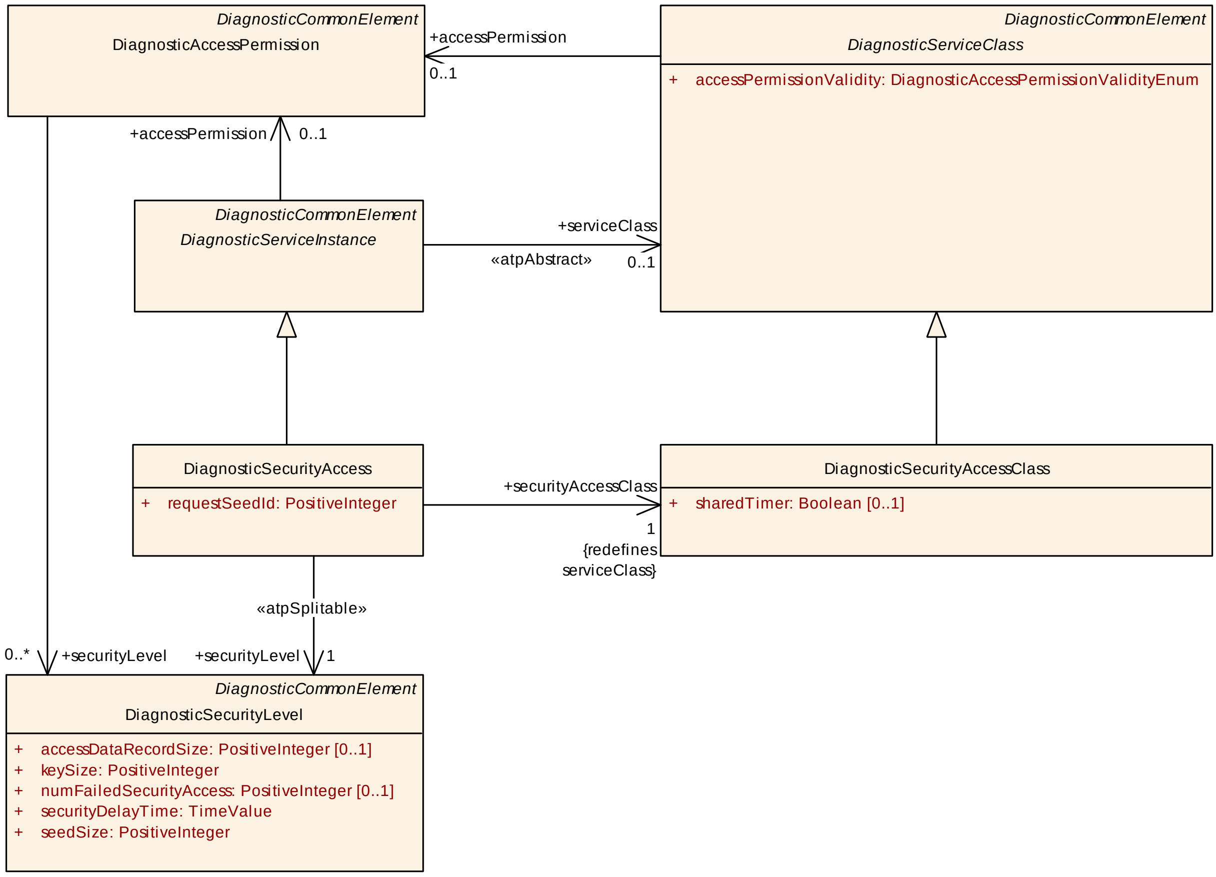


Figure 4.17: Refined modeling of the diagnostic security access

Class	**DiagnosticSecurityAccessClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess			
Note	This meta-class contains attributes shared by all instances of the "Security Access" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticSecurityAccesss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sharedTimer	Boolean	0..1	attr	Switch between separate or single shared timer instance and timer value. * True: use shared timer instance and timer value for all security access levels combined. * False: use separate timer instance and timer values for each security level. **Tags:** atp.Status=draft

Table 4.31: DiagnosticSecurityAccessClass

4.4 Custom Service Instance

Meta-class DiagnosticCustomServiceInstance can be used to define the existence of an instance of a custom diagnostic service.

Custom services can obviously not be configured using standardized attributes, but there is the ability to use `Sdg` in the context of `adminData` for this purpose.

There is no obligation for a given tool to be able to properly process the definition of the custom service instance.

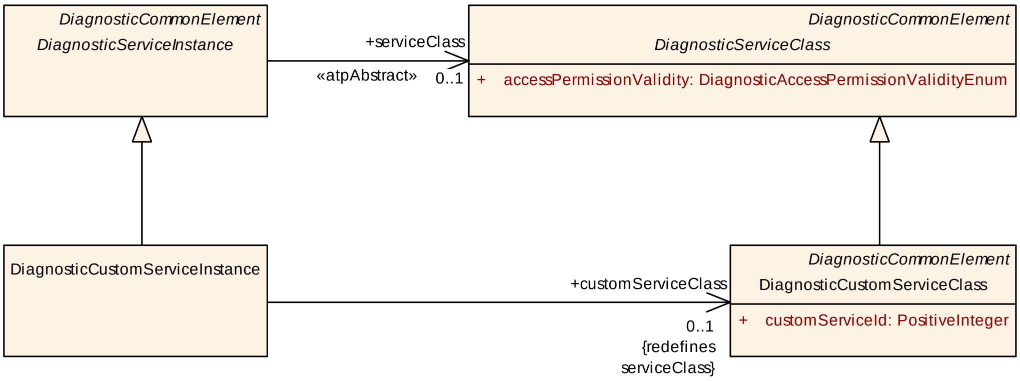


Figure 4.18: Modeling of custom service instance

Class	**DiagnosticCustomServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping::CustomServiceInstance			
Note	This meta-class has the ability to define an instance of a custom diagnostic service. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticCustomInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
customServiceClass	DiagnosticCustomServiceClass	0..1	ref	Reference to the corresponding DiagnosticCustomServiceClass. **Tags:** atp.Status=draft

Table 4.32: DiagnosticCustomServiceInstance

Class	**DiagnosticCustomServiceClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
Note	This represents the ability to define a custom diagnostic service class and assign an ID to it. Further configuration is not foreseen from the point of view of the diagnostic extract and consequently needs to be done on the level of ECUC. **Tags:** atp.recommendedPackage=DiagnosticCustomServiceClasses			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
customServiceId	PositiveInteger	1	attr	This attribute may only be used for the definition of custom services. The values shall not overlap with existing standardized service IDs.

Table 4.33: DiagnosticCustomServiceClass

4.5 DiagnosticProvidedDataMapping

[TPS_MANI_01230]{DRAFT} **Semantics of DiagnosticProvidedDataMapping**
⌈ The meta-class DiagnosticProvidedDataMapping does not seem to fulfill the condition for representing a mapping class because it only has one reference to a DiagnosticDataElement in the role dataElement.

However, the specific nature of this mapping is that the second element (the DiagnosticProvidedDataMapping.dataProvider) that is supposed to take place in the mapping cannot precisely be modeled as a single meta-class.

Therefore, there is no better way than to model the DiagnosticProvidedDataMapping.dataProvider by a NameToken. ⌋*()*

Class	**DiagnosticProvidedDataMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticProvidedDataMapping			
Note	This represents the ability to define the nature of a data access for a DiagnosticDataElement based on a data provider that cannot be modeled explicitly. **Tags:** atp.Status=draft atp.recommendedPackage=DataMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataElement	DiagnosticDataElement	0..1	ref	This represents the DiagnosticDataElement for which the access is further qualified by the DiagnosticProvidedData Mapping.dataProvider. **Tags:** atp.Status=draft
dataProvider	NameToken	1	attr	This represents the ability to further specify the data provider.

Table 4.34: DiagnosticProvidedDataMapping

Please note that the list of standardized values of attribute DiagnosticProvidedDataMapping.dataProvider is defined in the SWS Diagnostics [19].

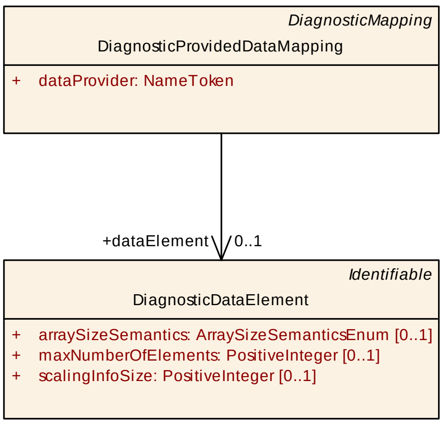


Figure 4.19: Modeling of DiagnosticProvidedDataMapping

5 System Design

5.1 Overview

A typical vehicle will most likely be equipped with ECUs developed on the AUTOSAR classic platform and ECUs developed on the AUTOSAR adaptive platform. The system design for the entire vehicle has therefore to cover all these ECUs.

The AUTOSAR model description supports the system design with the possibility to describe Software Components of both AUTOSAR Platforms that will be used in a System and even allows to indicate the service oriented communication between them if possible.

Especially when it come to the description of the communication behavior of AUTOSAR classic and adaptive ECUs in a harmonized way the notion of a System Design becomes a special focus point.

All the system design aspects have in common that they have to cope with both, AUTOSAR classic and adaptive. The basic design aspects of such interdisciplinary systems have to be already available in the AUTOSAR classic modeling approach because otherwise they would not be available to both worlds.

Thus it is straight forward to take the existing meta-class `System` as the starting point for the modeling of such mixed systems.

Class	**System**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. **Tags:** atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. atpVariation: In order to describe a product-line, all Fibex Elements can be optional. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects relevant in the System Description. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
pncVector Length	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).

▽

△

Class	System			
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.
rootSoftware Composition	RootSwComposition Prototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table 5.1: System

[constr_3366]{DRAFT} **System category for a system design description with Adaptive Platform and Classic Platform content** ⌈ The System element that contains design artifacts that are relevant for the Adaptive Platform and Classic Platform shall have the category SYSTEM_DESIGN_DESCRIPTION. ⌋*()*

There are use cases to exchange parts of such a SYSTEM_DESIGN_DESCRIPTION between different developer parties and therefore further system categories are supported by AUTOSAR.

A common approach is for example that the OEM provides a basis for designing an ECU, which is later advanced by the supplier. Therefore Classic AUTOSAR supports System categories like ECU_EXTRACT or ECU_SYSTEM_DESCRIPTION that have only a single ECU in scope.

Adaptive AUTOSAR is using the same approach. If an OEM wants to provide design artifacts that are relevant for the configuration of a single Machine all unnecessary information is stripped from the System with category SYSTEM_DESIGN_DESCRIPTION and a definition of the subsystem is provided.

[constr_3420]{DRAFT} **System category for a design description that has one single Adaptive Machine in scope** ⌈ The System element that contains design artifacts that are relevant for a single Adaptive Machine shall have the category MACHINE_DESIGN_EXTRACT. ⌋*()*

[constr_3421]{DRAFT} **Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT** ⌈ A System with the category MACHINE_DESIGN_EXTRACT is allowed to reference the following fibexElements:

- CommunicationCluster
- MachineDesign
- GlobalTimeDomain
- NmConfig
- SystemMapping that is allowed to contain only a PncMapping

⌋()

5.2 Specification of Communication System Structure

When the communication interaction is designed for a vehicle system the focus is put on the network and the connected ECUs. Whether a specific ECU connected to the network is implemented using AUTOSAR classic or AUTOSAR adaptive does not influence the major communication design.

But of course it is essential from a car manufacturer point of view whether a specific ECU will be implemented using AUTOSAR classic or adaptive. Thus already on system design level there is a need to specify the AUTOSAR Platform kind which shall be used to implement an ECU.

In AUTOSAR classic the element `EcuInstance` is used to define one ECU in the system design.

In AUTOSAR adaptive the element `Machine` is an entity which already represents a specific ECU Implementation with dedicated configurations for e.g. `Processor`s, `functionGroup`s. The `Machine` is a model entity which is not in the focus of communication designers and should not be used during system design.

Therefore the `MachineDesign` has been introduced in order to allow the communication system designer to define a placeholder for an adaptive ECU in the scope of the `System` (the `MachineDesign` corresponds to the `EcuInstance` of AUTOSAR classic).

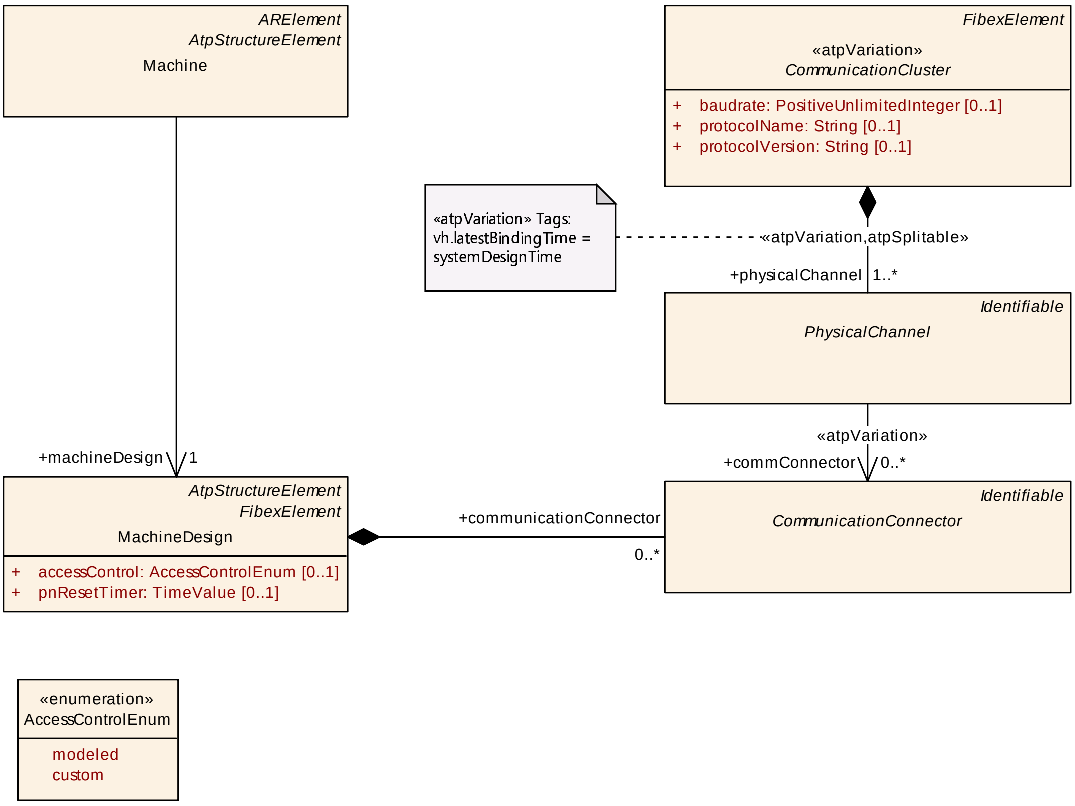


Figure 5.1: MachineDesign

Class	**MachineDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class represents the ability to define requirements on a Machine in the context of designing a system. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft atp.recommendedPackage=MachineDesigns			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
accessControl	AccessControlEnum	0..1	attr	This attribute defines how the access restriction to the Service Instance is defined.
communication Connector	Communication Connector	*	aggr	This aggregation defines the network connection of the machine. **Tags:** atp.Status=draft
pnResetTimer	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests.
serviceDiscover Config	ServiceDiscovery Configuration	*	aggr	Set of service discovery configuration settings that are defined on the machine for individual Communication Connectors. **Tags:** atp.Status=draft

Table 5.2: MachineDesign

[TPS_MANI_03209]{DRAFT} **The meaning of MachineDesign.accessControl**
⌈ The MachineDesign.accessControl defines whether the access control is defined by AUTOSAR means in the Application Design with receiverCapability (see [TPS_MANI_01106]) and senderCapability (see [TPS_MANI_01107]) or by a custom lists that are created by a non-AUTOSAR process. ⌋*(RS_MANI_00034)*

Enumeration	**AccessControlEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	This enumeration describes the options for the definition of access restriction to resources. **Tags:** atp.Status=draft
Literal	***Description***
custom	The access restriction to the resource is defined by a non-AUTOSAR process. **Tags:** atp.EnumerationValue=1
modeled	The access restriction to the resource is modeled in the AUTOSAR Application Design model or the AUTOSAR Deployment model. **Tags:** atp.EnumerationValue=0

Table 5.3: AccessControlEnum

5.2.1 Network connection

One of the most prominent information defined in the context of the MachineDesign is the network connectivity. Since the *AUTOSAR adaptive platform* focuses on the usage of Ethernet for communication, this boils down to the specification of IP addresses.

Specifically, the basic definition of the connectivity of a MachineDesign is created by aggregating the abstract base-class CommunicationConnector in the role communicationConnector. The specific subclass of CommunicationConnector that is used in this context is the EthernetCommunicationConnector.

The EthernetCommunicationConnector is used to connect the MachineDesign with a VLAN that is represented in AUTOSAR by a EthernetPhysicalChannel that is part of an EthernetCluster.

Class	***PhysicalChannel*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	This element represents a physical connection (in case of CAN, FlexRay, LIN) or a logical connection (VLAN in case of Ethernet) between communicating devices.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanPhysicalChannel*, EthernetPhysicalChannel, FlexrayPhysicalChannel, LinPhysicalChannel, UserDefinedPhysicalChannel			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	*PhysicalChannel* (abstract)			
comm Connector	Communication Connector	*	ref	Reference to the ECUInstance via a Communication Connector to which the channel is connected. atpVariation: Variable assignment of Physical Channels to different CommunicationConnectors is expressed with this variation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild

Table 5.4: PhysicalChannel

Class	«atpVariation» **EthernetCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet-specific cluster attributes. **Tags:** atp.ManifestKind=MachineManifest atp.recommendedPackage=CommunicationClusters			
Base	*ARObject*, *CollectableElement*, *CommunicationCluster*, *FibexElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
couplingPort Connection	CouplingPort Connection	*	aggr	Specification of connections between CouplingElements and EcuInstances. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=couplingPortConnection, variation Point.shortLabel vh.latestBindingTime=postBuild
couplingPort SwitchoffDelay	TimeValue	0..1	attr	Switch off delay for CouplingPorts in seconds. It denotes the delay of switching off couplingPorts after the request to switch off a couplingPort was issued. (e.g. switch off of Ethernet switch ports).
macMulticast Group	MacMulticastGroup	*	aggr	MacMulticastGroup that is defined for the Subnet (EthernetCluster).

Table 5.5: EthernetCluster

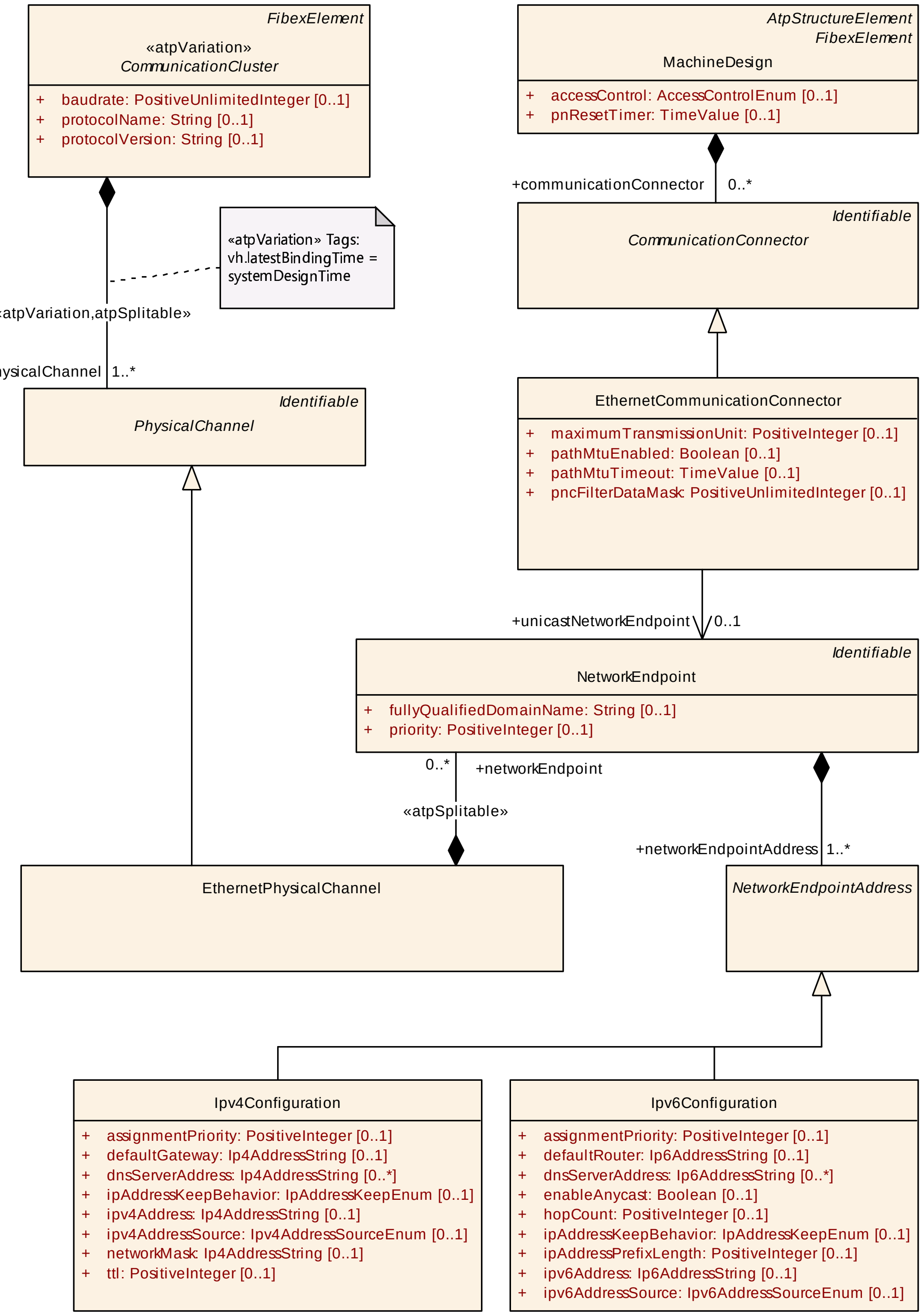


Figure 5.2: Network connection of a MachineDesign

[constr_3320]{DRAFT} **Aggregation of CommunicationConnector by MachineDesign** ⌈ Meta-Class MachineDesign shall only aggregate EthernetCommunicationConnectors in the role communicationConnector. No other subclass of CommunicationConnector shall appear in this aggregation. ⌋*()*

The canonical way to specify an IP address is the modeling of a NetworkEndpoint, referenced from an EthernetCommunicationConnector that is aggregated by MachineDesign in the role communicationConnector.

In addition to the IP address, the NetworkEndpoint may have a *Fully Qualified Domain Name* and a priority.

Class	NetworkEndpoint			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	The network endpoint defines the network addressing (e.g. IP-Address or MAC multicast address). **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
fullyQualified DomainName	String	0..1	attr	Defines the fully qualified domain name (FQDN) e.g. some.example.host.
ipSecConfig	IPSecConfig	0..1	aggr	Optional IPSec configuration that provides security services for IP packets. **Tags:** atp.Status=draft
network Endpoint Address	NetworkEndpoint Address	1..*	aggr	Definition of a Network Address. **Tags:** xml.name Plural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.

Table 5.6: NetworkEndpoint

More precisely, the particular IP address is configured by means of the aggregation of Ipv4Configuration or Ipv6Configuration in the role networkEndpointAddress.

The NetworkEndpoint is aggregated by the EthernetPhysicalChannel that in turn is aggregated by the EthernetCluster.

Please note that the reference commConnector from the EthernetPhysicalChannel to the CommunicationConnector is optional although the lower multiplicity in the model is 1. The multiplicity of 1 is related to AUTOSAR Classic Platform and will be changed in future.

[TPS_MANI_03052]{DRAFT} **Static IPv4 configuration** ⌈ If the value of attribute ipv4AddressSource of meta-class Ipv4Configuration is set to Ipv4AddressSourceEnum.fixed then the ipv4Address defines the static IPv4 Address. ⌋*(RS_MANI_00018)*

Class	Ipv4Configuration			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 4 (IPv4) configuration.			
Base	*ARObject*, *NetworkEndpointAddress*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
assignment Priority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.

▽

△

Class	Ipv4Configuration			
defaultGateway	Ip4AddressString	0..1	attr	IP address of the default gateway.
dnsServer Address	Ip4AddressString	*	attr	IP addresses of preconfigured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
ipAddressKeep Behavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipv4Address	Ip4AddressString	0..1	attr	IPv4 Address. Notation: 255.255.255.255. The IP Address shall be declared in case the ipv4Address Source is FIXED and thus no auto-configuration mechanism is used.
ipv4Address Source	Ipv4AddressSource Enum	0..1	attr	Defines how the node obtains its IP address.
networkMask	Ip4AddressString	0..1	attr	Network mask. Notation 255.255.255.255
ttl	PositiveInteger	0..1	attr	Lifespan of data (0..255). The purpose of the TimeToLive field is to avoid a situation in which an undeliverable datagram keeps circulating on a system.

Table 5.7: Ipv4Configuration

Enumeration	**Ipv4AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines how the node obtains its IPv4-Address.
Literal	***Description***
autoIp	AutoIP is used to dynamically assign IP addresses at device startup. **Tags:** atp.EnumerationValue=0
autoIp_doip	Linklocal IPv4 Address Assignment using DoIP Parameters **Tags:** atp.EnumerationValue=2
dhcpv4	DHCP is a service for the automatic IP configuration of a client. **Tags:** atp.EnumerationValue=3
fixed	The IP Address shall be declared manually. **Tags:** atp.EnumerationValue=4

Table 5.8: Ipv4AddressSourceEnum

[TPS_MANI_03053]{DRAFT} **Static IPv6 configuration** ⌈ If the value of attribute `ipv6AddressSource` of meta-class `Ipv6Configuration` is set to `Ipv6AddressSourceEnum.fixed` then the `ipv6Address` defines the static IPv6 Address. ⌋*(RS_MANI_00018)*

Class	**Ipv6Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 6 (IPv6) configuration.			
Base	*ARObject*, *NetworkEndpointAddress*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	**Ipv6Configuration**			
assignment Priority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.
defaultRouter	Ip6AddressString	0..1	attr	IP address of the default router.
dnsServer Address	Ip6AddressString	*	attr	IP addresses of pre configured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
enableAnycast	Boolean	0..1	attr	This attribute is used to enable anycast addressing (i.e. to one of multiple receivers).
hopCount	PositiveInteger	0..1	attr	The distance between two hosts. The hop count n means that n gateways separate the source host from the destination host (Range 0..255)
ipAddressKeep Behavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipAddressPrefix Length	PositiveInteger	0..1	attr	IPv6 prefix length defines the part of the IPv6 address that is the network prefix.
ipv6Address	Ip6AddressString	0..1	attr	IPv6 Address. Notation: FFFF:...:FFFF. The IP Address shall be declared in case the ipv6Address Source is FIXED and thus no auto-configuration mechanism is used.
ipv6Address Source	Ipv6AddressSource Enum	0..1	attr	Defines how the node obtains its IP address.

Table 5.9: Ipv6Configuration

Enumeration	**Ipv6AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines how the node obtains its IPv6-Address.
Literal	*Description*
dhcpv6	DHCP is a service for the automatic IP configuration of a client. **Tags:** atp.EnumerationValue=0
fixed	The IP Address shall be declared manually. **Tags:** atp.EnumerationValue=1
linkLocal	LinkLocal is intended only for communications within the segment of a local network (a link) or a point-to-point connection that a host is connected to. **Tags:** atp.EnumerationValue=2
linkLocal_doip	Linklocal IPv6 Address Assignment using DoIP Parameters **Tags:** atp.EnumerationValue=3
router Advertisement	IPv6 Stateless Autoconfiguration. **Tags:** atp.EnumerationValue=4

Table 5.10: Ipv6AddressSourceEnum

Enumeration	**IpAddressKeepEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines the behavior after a dynamic IP address has been assigned.
Literal	**Description**
forget	After a dynamic IP address has been assigned just use it for this session. **Tags:** atp.EnumerationValue=0
storePersistently	After a dynamic IP address has been assigned store the address persistently. **Tags:** atp.EnumerationValue=1

Table 5.11: IpAddressKeepEnum

5.2.2 Securing Communication with IPsec

IPsec is a protocol suite that provides cryptographic protection for IP datagrams in IPv4 and IPv6 network packets.

IPsec uses a security association to specify security properties that are shared between the communicating parties. The security association defines a relationship between two or more parties and determines which security services will be used to communicate securely. In other words the security association serves as a “contract” between the different devices.

A single security association protects data in one communication direction. Two security associations must be present to secure traffic in both directions. Each security association can provide encryption, data integrity and data authentication.

In addition the senders and receivers of IP datagrams can determine the required protection for an IP packet according to IPsec security policies. These are rules that define how datagrams are processed that are received by a device. For example, security policies are used to decide if a particular packet needs to be dropped or needs to be processed by IPsec.

[TPS_MANI_03203]{DRAFT} **Configuration of IPsec** ⌈ The `IPSecConfig` meta-class that is aggregated by a `NetworkEndpoint` in the role `ipSecConfig` provides the ability to define IPsec settings that are necessary to configure IPsec security associations and IPsec security policies. ⌋*(RS_MANI_00036)*

[TPS_MANI_03204]{DRAFT} **Definition of `IPSecRules`** ⌈ The `IPSecConfig` meta-class may contain one or several `IPSecRule`s. Each `IPSecRule` defines the network connection that is monitored by IPsec by defining the local endpoint and the remote endpoint. Each endpoint is defined by the IP Address and the Tcp/Udp Port. The communication direction for which the `IPSecRule` is valid is defined by the `direction` attribute. ⌋*(RS_MANI_00036)*

Class	IPSecConfig			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	IPSec is a protocol that is designed to provide "end-to-end" cryptographically-based security for IP network connections. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
ipSecRule	IPSecRule	*	aggr	IPSec rules and filters that are defined in the IPSecConfig for a specific NetworkEndpoint. **Tags:** atp.Status=draft

Table 5.12: IPSecConfig

Class	IPSecRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This element defines an IPSec rule that describes communication traffic that is monitored, protected and filtered. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
ahCipherSuite Name	String	*	attr	AH (Authentication Header) algorithm to be used for the connection, e.g. sha1-sha256-modp1024.
connectionType	IPSecConnectionType Enum	1	attr	This attribute defines the type of the connection.
direction	Communication DirectionType	0..1	attr	This attribute defines the direction in which the traffic is monitored. If this attribute is not set a bidirectional traffic monitoring is assumed.
espCipherSuite Name	String	*	attr	ESP (Encapsulating Security Payload) algorithm that provides encryption and optional authentication for the connection, e.g. aes128-sha256.
ike Authentication Method	IkeAuthentication MethodEnum	0..1	attr	This attribute defines the IKE authentication method that is used locally and is expected on the remote side.
rekeyInterval	TimeValue	0..1	attr	This attribute provides the information how long (in seconds) the Security Association (SA) defined by this IPSecRule shall be used.
remoteIp Address	NetworkEndpoint	*	ref	Definition of the remote NetworkEndpoint. With this reference the connection between the local Network Endpoint and the remote NetworkEndpoint is described on which the traffic is monitored. **Tags:** atp.Status=draft
tcpLocalPort	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring to tcp and a defined local port. LocalPort = 0 means ANY.
tcpRemotePort	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring to tcp and a defined remote port. LocalPort = 0 means ANY.
udpLocalPort	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring to udp and a defined local port. LocalPort = 0 means ANY.

▽

△

Class	**IPSecRule**			
udpRemotePort	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring to udp and a defined remote port. LocalPort = 0 means ANY.

Table 5.13: IPSecRule

Enumeration	**IkeAuthenticationMethodEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign
Note	This enumeration describes the supported IKE authentication methods. **Tags:** atp.Status=draft
Literal	***Description***
dsa	Digital Signature Authentication **Tags:** atp.EnumerationValue=2
psk	Pre-shared key authentication **Tags:** atp.EnumerationValue=1

Table 5.14: IkeAuthenticationMethodEnum

Enumeration	**IPSecConnectionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign
Note	This enumeration describes the supported IPSec connection types. **Tags:** atp.Status=draft
Literal	***Description***
drop	Signifying that packets should be discarded **Tags:** atp.EnumerationValue=3
passthrough	Signifying that no IPsec processing should be done at all. **Tags:** atp.EnumerationValue=2
reject	Signifying that packets should be discarded and a diagnostic ICMP returned. **Tags:** atp.EnumerationValue=4
transport	Signifying that the IPSec transport mode is used. With the transport mode the original IP header is retained and only the IP payload and ESP trailer is encrypted. **Tags:** atp.EnumerationValue=1
tunnel	Signifying that the IPSec tunnel mode is used. With tunnel mode, the entire original IP packet is protected by IPSec. This means IPSec wraps the original packet, encrypts it, adds a new IP header and sends it to the other side. **Tags:** atp.EnumerationValue=0

Table 5.15: IPSecConnectionTypeEnum

[TPS_MANI_03205]{DRAFT} **IPsec connection type** ⌈ The IPSecRule.connectionType attribute defines how IP packets are handled that are going over the network connection defined by the IPSecRule. In detail it defines whether the IP packet is processed by IPsec or not. If processed it defines whether the IPsec transport mode or the IPsec tunnel mode is used. ⌋*(RS_MANI_00036)*

IPsec can be configured to operate in two different modes, Tunnel and Transport mode. With tunnel mode, the entire IP packet is protected by IPsec. IPsec wraps the original packet, encrypts it and adds a new IP header to it. The tunnel mode is most commonly used between VPN gateways and the IP addresses of the newly added outer IP header are that of the VPN Gateways. In other words the traffic between the two VPN Gateways is protected and each gateway acts as a proxy for the hosts behind it.

The transport mode provides the protection of the Data Payload of the IP datagram with an AH or ESP header. The IP Header remains the same and IPsec inserts its header between the IP header and the upper level headers. The IPsec transport mode can be used when securing traffic between two hosts or between a host and a VPN gateway.

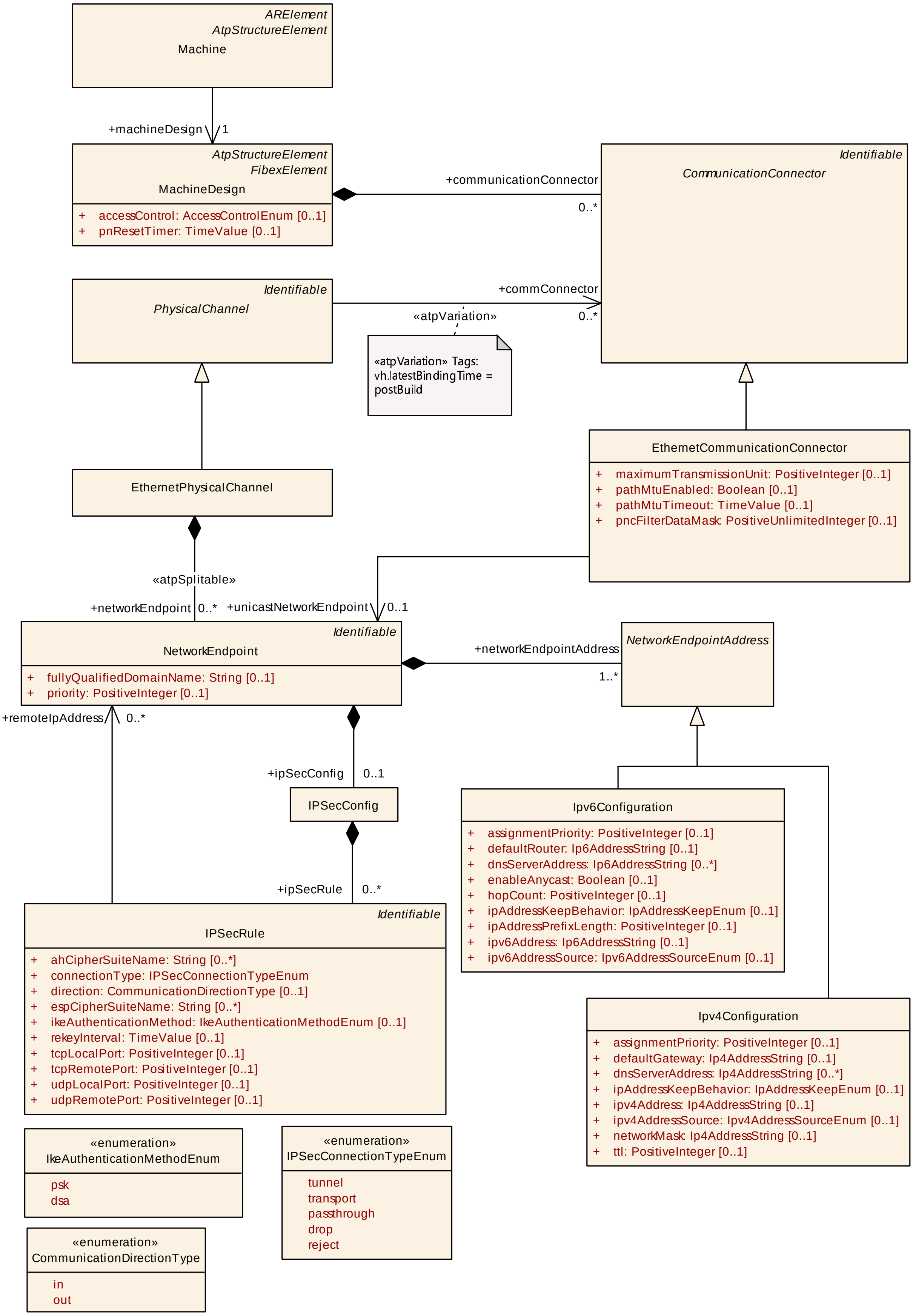


Figure 5.3: IPsec configuration model

IPsec uses two protocols:

- AH - Authentication Header
- ESP - Encapsulating Security Payload

The AH protocol provides a mechanism for authentication only and authenticates the entire IP packet, including the outer IP header.

The ESP protocol provides data confidentiality (encryption) and authentication (data integrity, data origin authentication, and replay protection).

When ESP is used in transport mode, the IP payload is encrypted and the original IP header is moved to the front of the message. The ESP header is inserted after the IP header and is signed together with the IP payload. The original IP header remains unprotected.

When ESP is used in tunnel mode a new IP Header is created and the ESP header is added in front of the original IP Packet. The entire original IP packet is encrypted and signed in this mode.

[TPS_MANI_03206]{DRAFT} **IPsec AH and ESP protocol configuration** ⌈ In the `IPSecRule` it is possible to define the IPsec protocol that shall be used to protect IP packets that are going over the defined network connection. The usage of the attributes `ahCipherSuiteName` and `espCipherSuiteName` defines whether AH or ESP is used. In addition the attributes define the AH and ESP algorithms that are used. ⌋*(RS_MANI_00036)*

[TPS_MANI_03207]{DRAFT} **IPsec Internet Key Exchange protocol configuration** ⌈ In the `IPSecRule` it is possible to define how IKE protocol authenticates other party. The usage of the attribute `ikeAuthenticationMethod` defines whether Digital Signature Authentication or pre-shared key is used. In addition the attribute `rekeyInterval` provides information how long a Security Association (SA) shall be used. ⌋ *(RS_MANI_00036)*

[TPS_MANI_03208]{DRAFT} **Protection of `AdaptivePlatformServiceInstance` by IPsec** ⌈ To describe the protection of an `AdaptivePlatformServiceInstance` by IPsec the `AdaptivePlatformServiceInstance` needs to be mapped by a `ServiceInstanceToMachineMapping` to an `EthernetCommunicationConnector` that points with the `unicastNetworkEndpoint` to a `NetworkEndpoint` that aggregates the `IPSecConfig` that in turn describes IPsec Security Associations. ⌋*(RS_MANI_00036)*

Please note that IP Multicast protection by IPsec is not supported. It is by intention not possible to model the IPsec protection of IP Multicast communication since the IP Multicast address is defined in the `SomeipServiceInstanceToMachineMapping` by the two attributes `ipv4MulticastIpAddress` and `ipv6MulticastIpAddress`. The `NetworkEndpoint` element is used for description of IP Unicast Endpoints only. This means that only the IP Unicast communication of an `AdaptivePlatformServiceInstance` that is described according to [TPS_MANI_03208] will be protected by IPsec.

5.2.3 Service Discovery Configuration

Service Discovery messages are exchanged between network nodes to announce and to discover available service instances. This chapter describes the configuration that is necessary to exchange service discovery messages for supported middleware transport layers.

Class	***ServiceDiscoveryConfiguration*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	Service Discovery configuration settings for the middleware transport layer. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*			
Subclasses	SomeipServiceDiscovery			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 5.16: ServiceDiscoveryConfiguration

5.2.3.1 SOME/IP Service Discovery Configuration

[TPS_MANI_03064]{DRAFT} **SOME/IP Service Discovery message exchange configuration** ⌈ `ProvidedServiceInstance`s are announced in SOME/IP by the server with multicast addressing on a `VLAN` to a specifically designated IP multicast address (`SomeipServiceDiscovery.multicastSdIpAddress`) at a specific UDP port number (`SomeipServiceDiscovery.someipServiceDiscoveryPort`). ⌋*(RS_MANI_00019)*

[constr_5045]{DRAFT} **Only one `SomeipServiceDiscovery` configuration per VLAN is allowed** ⌈ Only a single `NetworkEndpoint` on an `EthernetPhysicalChannel` (VLAN) is allowed to be referenced by a `SomeipServiceDiscovery` element in the role `multicastSdIpAddress`. ⌋*()*

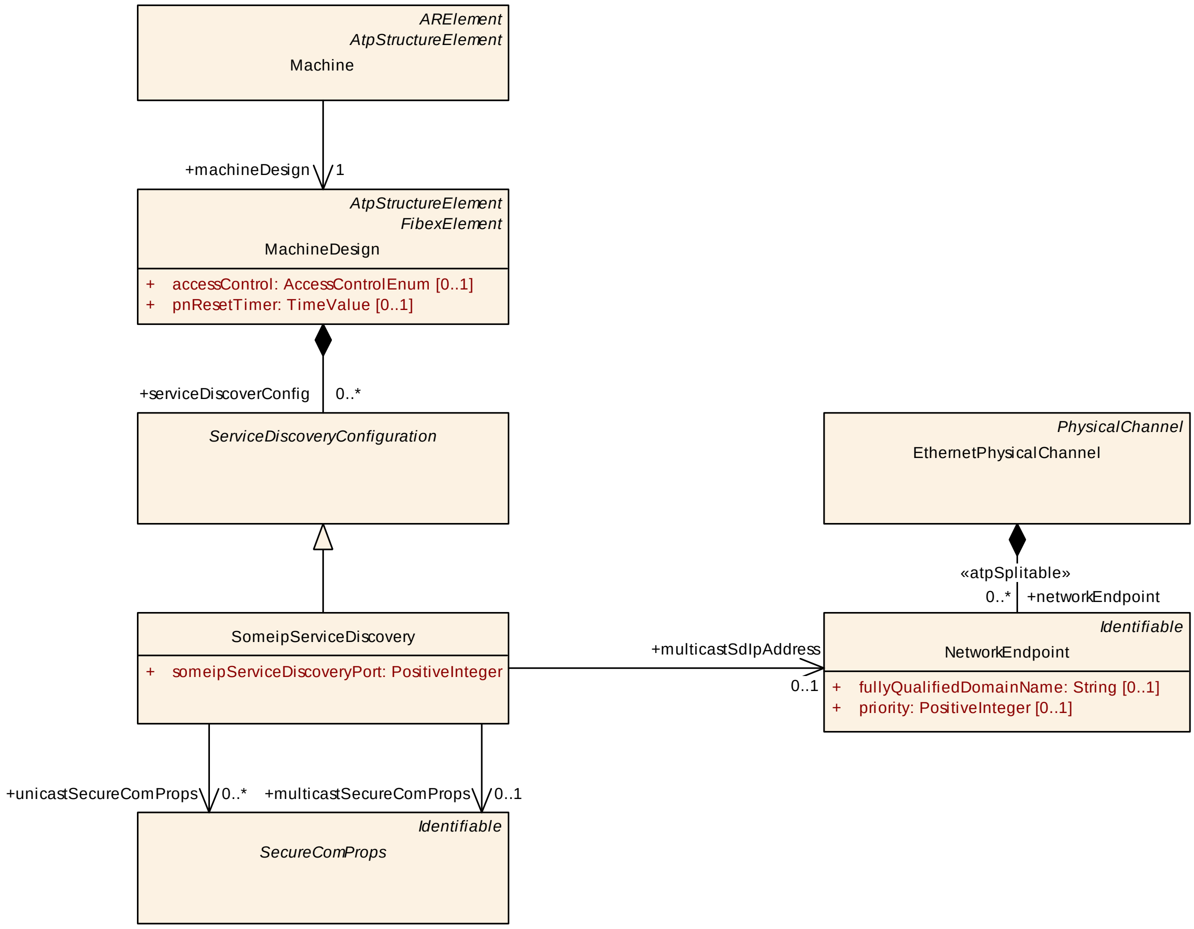


Figure 5.4: SOME/IP Service Discovery Configuration

Class	**SomeipServiceDiscovery**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This meta-class represents a specialization of the generic service discovery for the SOME/IP case. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *ServiceDiscoveryConfiguration*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
multicastSdIp Address	NetworkEndpoint	0..1	ref	This reference identifies the multicast IP address used for service discovery. **Tags:** atp.Status=draft
multicastSecure ComProps	SecureComProps	0..1	ref	Reference to a communication security protocol and its configuration settings that will provide communication security for Service Discovery messages that are transmitted using multicast, e.g. FindService message. **Tags:** atp.Status=draft
someipService DiscoveryPort	PositiveInteger	1	attr	This attribute represents the port number reserved for service discovery.

▽

△

Class	**SomeipServiceDiscovery**			
unicastSecureComProps	SecureComProps	*	ref	Reference to a communication security protocol and its configuration settings that will provide communication security for Service Discovery messages that are transmitted using unicast, e.g. OfferService as answer to a FindService message. . **Tags:** atp.Status=draft

Table 5.17: SomeipServiceDiscovery

The SomeipServiceDiscovery is able to reference SecureComProps to define and to configure a security protocol that will be provide communication security for Service Discovery messages. For Service Discovery messages that will be transmitted to a designated multicast IP address the protection is defined by the SecureComProps that is referenced in the role multicastSecureComProps. For unicast Service Discovery messages different credentials may be used for the different ECU pairs. Therefore a list of SecureComProps is aggregated in the role unicastSecureComProps.

5.2.4 Partial Network

AUTOSAR supports power saving during vehicle operation time with the partial networking mechanism. This mechanism allows to shut down and startup the bus communication interfaces of groups of ECUs (Partial Network Cluster) during normal bus communication. On the VFB Level Partial Networks are represented by Virtual Function Clusters and are described with PortGroups. The Virtual Function Cluster groups the communication necessary to realize one or more vehicle functions that can become activated/deactivated during normal vehicle operation. The Virtual Function Clusters are mapped onto Partial Network Clusters.

[TPS_MANI_03224]{DRAFT} **Modeling of a Partial Network Cluster** ⌈ A Partial Network Cluster is modeled with the PncMapping element and is identified by the pncIdentifier. The PncMapping defines the collection of AdaptivePlatformServiceInstances that are participating in the partial network with the PncMapping.serviceInstance reference. ⌋*(RS_MANI_00062)*

[TPS_MANI_03225]{DRAFT} **References to vlans in PncMapping** ⌈ An EthernetCommunicationConnector may be referenced directly by a given PncMapping in the role physicalChannel and also by a ServiceInstanceToMachineMapping that maps an AdaptivePlatformServiceInstance to a EthernetCommunicationConnector that in turn is referenced by the same PncMapping in the role serviceInstance. ⌋*(RS_MANI_00062)*

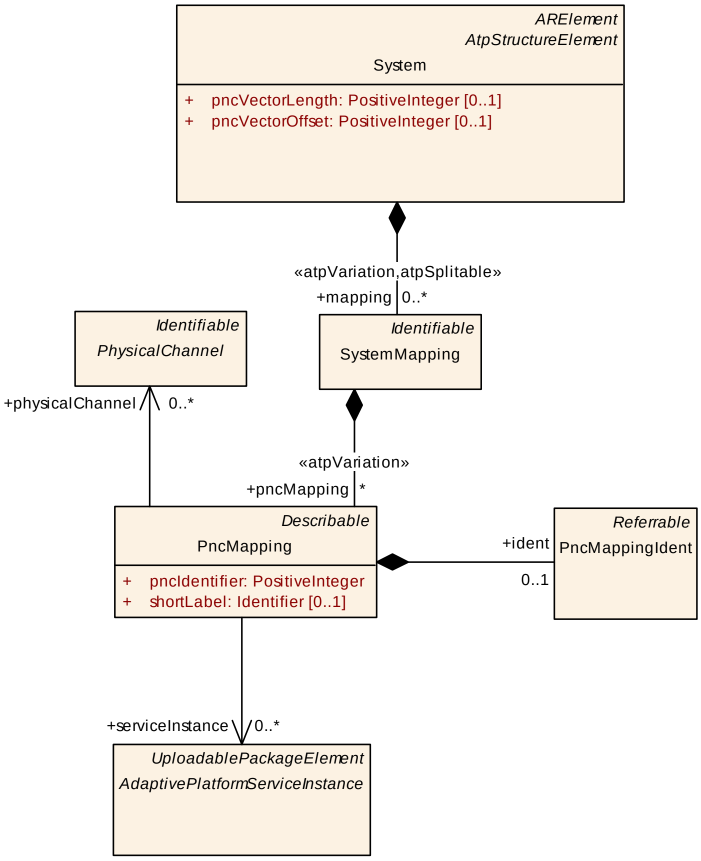


Figure 5.5: PncMapping with collection of ServiceInstances that are participating in the Partial Network Cluster

Class	**SystemMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The system mapping aggregates all mapping aspects that are relevant in the System Description.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
pncMapping	PncMapping	*	aggr	Mappings between Virtual Function Clusters and Partial Network Clusters. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime

Table 5.18: SystemMapping

Class	**PncMapping**
Package	M2::AUTOSARTemplates::SystemTemplate::PncMapping
Note	Describes a mapping between one or several Virtual Function Clusters onto Partial Network Clusters. A Virtual Function Cluster is realized by a PortGroup. A Partial Network Cluster is realized by one or more ServiceInstances.
Base	*ARObject*, *Describable*

▽

△

Class	PncMapping			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
ident	PncMappingIdent	0..1	aggr	This adds the ability to become referrable to PncMapping.
physical Channel	PhysicalChannel	*	ref	This reference maps the partial network to a communication channel.
pncIdentifier	PositiveInteger	1	attr	Identifer of the Partial Network Cluster. This number represents the absolute bit position of this Partial Network Cluster in the NM Pdu.
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	Reference to ServiceInstances that are participating in a Partial Network Cluster. **Tags:** atp.Status=draft
shortLabel	Identifier	0..1	attr	This attribute specifies an identifying shortName for the PncMapping. It shall be unique in the System scope.
vfc	PortGroup	*	iref	Virtual Function Cluster to be mapped onto a Partial Network Cluster. This reference is optional in case that the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy systems.

Table 5.19: PncMapping

5.3 Specification of Application Software System Structure

The root element of a System Design model is the `System` element that is already known from the AUTOSAR classic platform. The `System` aggregates the `RootSwCompositionPrototype` that represents the top-level-composition of all software components that are available in a given system.

[TPS_MANI_03110]{DRAFT} **Allowed components in system description with `category` `SYSTEM_DESIGN_DESCRIPTION`.** ⌈ `SwComponentPrototype`s nested inside the `CompositionSwComponentType` that is referenced by the `RootSwCompositionPrototype` of a `System` with `category` `SYSTEM_DESIGN_DESCRIPTION` are allowed to be of any `SwComponentType` that is supported by Classic or by Adaptive Autosar. ⌋*(RS_MANI_00026)*

Class	RootSwCompositionPrototype
Package	M2::AUTOSARTemplates::SystemTemplate
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within a given System. According to the use case of the System, this may for example be the a more or less complete VFB description, the software of a System Extract or the software of a flat ECU Extract with only atomic SWCs. Therefore the RootSwComposition will only occasionally contain all atomic software components that are used in a complete VFB System. The OEM is primarily interested in the required functionality and the interfaces defining the integration of the Software Component into the System. The internal structure of ▽

▽

△

Class	RootSwCompositionPrototype			
	△ such a component contains often substantial intellectual property of a supplier. Therefore a top-level software composition will often contain empty compositions which represent subsystems. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including PortPrototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.			
Base	*ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
software Composition	CompositionSw ComponentType	1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system **Stereotypes:** isOfType

Table 5.20: RootSwCompositionPrototype

If a Software Component communicates over the service oriented communication and provides or requires a ServiceInterface the opposite communication end is not always known upfront. In the System with category SYSTEM_DESIGN_DESCRIPTION a System Designer may want to indicate the service oriented communication between endpoints if it is already known at the System Design time.

[TPS_MANI_03114]{DRAFT} **Usage of AssemblySwConnectors in the System Design model** ⌈ In the System with category SYSTEM_DESIGN_DESCRIPTION it is allowed to indicate the service oriented communication between two communication endpoints by AssemblySwConnectors if the required RPortPrototype is searching for a specific service instance, i.e. if the RPortPrototypeProps.searchBehavior is set to searchForId.

If the searchBehavior is set to searchForAny the AssemblySwConnector shall not be used to connect this RPortPrototype. ⌋*(RS_MANI_00026)*

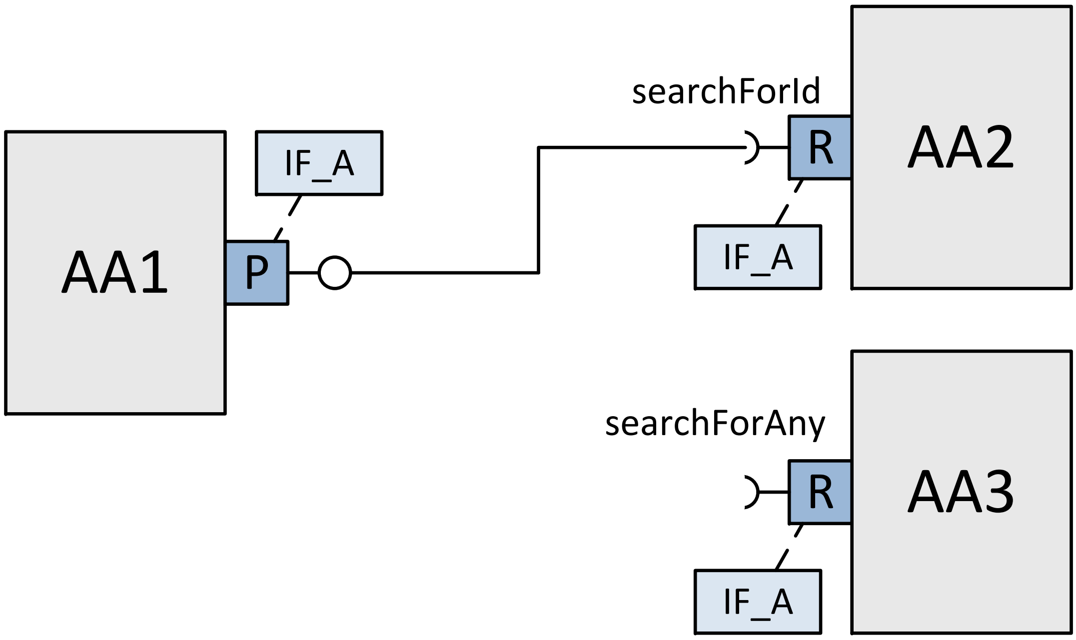


Figure 5.6: Example for Assembly connectors in System Design model

5.4 Modeling of service oriented communication between Classic and Adaptive platform

AUTOSAR classic platform does not support `ServiceInterface`s yet but provides the possibility to communicate in a service oriented way over SOME/IP. To mimic a `ServiceInterface` in the classic platform any combination of `ClientServerInterface`s, `SenderReceiverInterface`s or `TriggerInterface`s may be used to describe a service to which later a SOME/IP Service Id is assigned.

To simplify the description of the service oriented communication between Classic and Adaptive Software components in a System design model the `InterfaceMapping` was introduced that allows to map elements of `PortInterface`s of the Classic Platform to a single `ServiceInterface` of the Adaptive Platform.

Class	**InterfaceMappingSet**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign
Note	This meta-class represents the ability to aggregate a collection of InterfaceMappings. **Tags:** atp.Status=draft atp.recommendedPackage=InterfaceMappingSets
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*

▽

△

Class	InterfaceMappingSet			
Attribute	Type	Mul.	Kind	Note
interface Mapping	InterfaceMapping	*	aggr	Mapping of a ServiceInterface of the Adaptive Platform to PortInterface elements of the Classic Platform. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName,variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=systemDesignTime

Table 5.21: InterfaceMappingSet

Class	InterfaceMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class collects the mappings of elements of a single ServiceInterface to PortInterface elements of the AUTOSAR Classic Platform. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	Type	Mul.	Kind	Note
eventMapping	EventMapping	*	aggr	Mapping of a VariableDataPrototype in a SenderReceiver Interface to an Event in a ServiceInterface. **Tags:** atp.Status=draft
fieldMapping	FieldMapping	*	aggr	Mapping of a Field in a ServiceInterface to ClientServer Operations that represent the getter and setter methods and to a VariableDataPrototype that represents the notifier in the Field. **Tags:** atp.Status=draft
fireAndForget Mapping	FireAndForgetMapping	*	aggr	Mapping of a Fire&Forget Method that is located in a ServiceInterface to a VariableDataPrototype in a Sender ReceiverInterface or to a Trigger in a TriggerInterface. **Tags:** atp.Status=draft
methodMapping	MethodMapping	*	aggr	Mapping of a ClientServerOperation in a ClientServer Interface to a Method in a ServiceInterface. **Tags:** atp.Status=draft

Table 5.22: InterfaceMapping

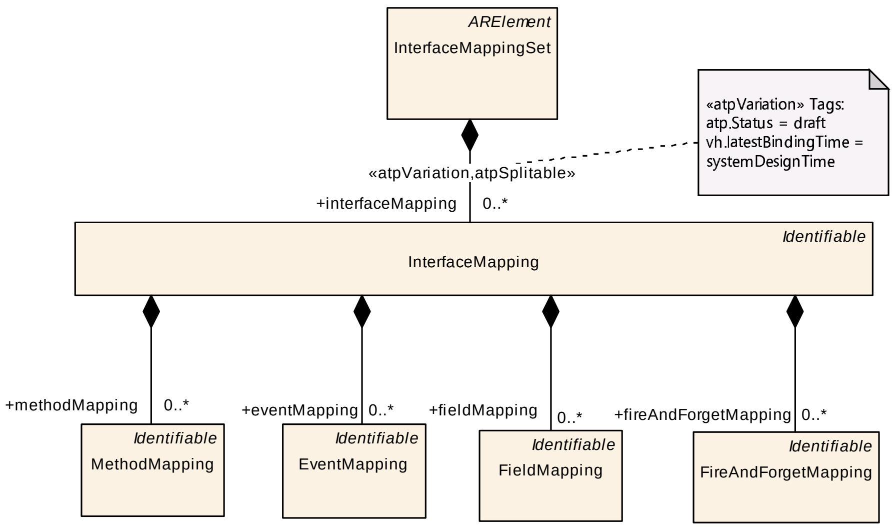


Figure 5.7: InterfaceMapping Overview

[constr_3370]{DRAFT} **InterfaceMapping shall map all elements of a single ServiceInterface** ⌈ The mappings that are included in an InterfaceMapping shall map all elements of a single ServiceInterface (i.e. fields, events, methods) to PortInterface elements of the classic platform. ⌋*()*

Figure 5.8 shows a possible System Design modeling approach where Adaptive Applications are communicating in a service oriented way over SOME/IP with classic Software Components. SWC_1 requires a ClientServerInterface with a ClientServerOperation and a SenderReceiverInterface with a VariableDataPrototype. SWC_2 requires a SenderReceiverInterface with with a VariableDataPrototype. The three Interfaces are mapped by a InterfaceMapping to a single ServiceInterface IF_C. On the other side the Adaptive Applications AA1 and AA2 provide ServiceInterfaces IF_A and IF_B that are composed by a ServiceInterfaceMapping to IF_C.

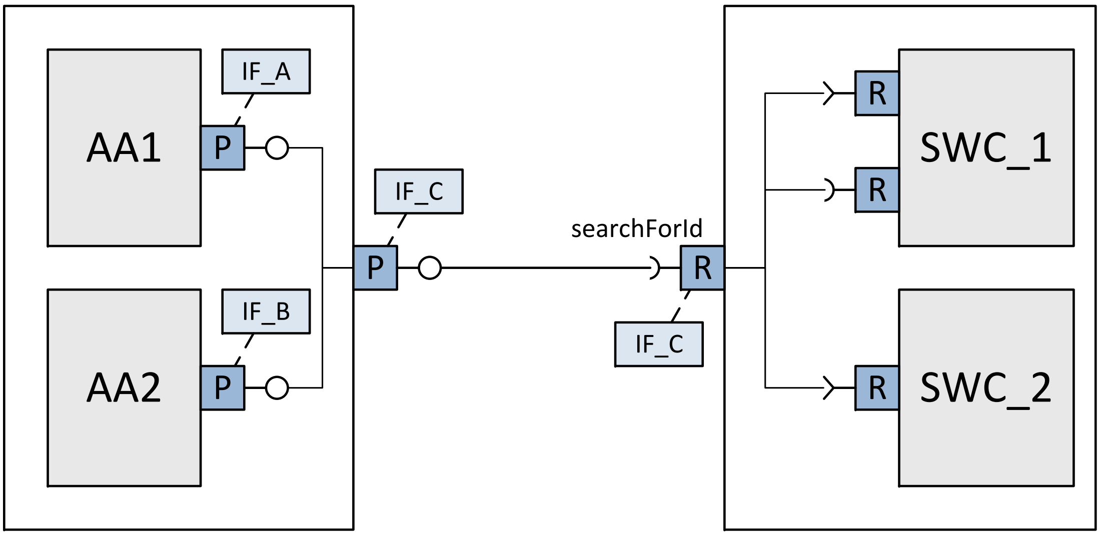


Figure 5.8: Example for a modeling of Service Oriented communication between Adaptive Applications and Software Components of the Classic Platform

5.4.1 MethodMapping

[TPS_MANI_03111]{DRAFT} **Mapping between method and operation** ⌈ The mapping between a method located in a ServiceInterface and a operation located in a ClientServerInterface is provided by the class MethodMapping. ⌋
(RS_MANI_00026)

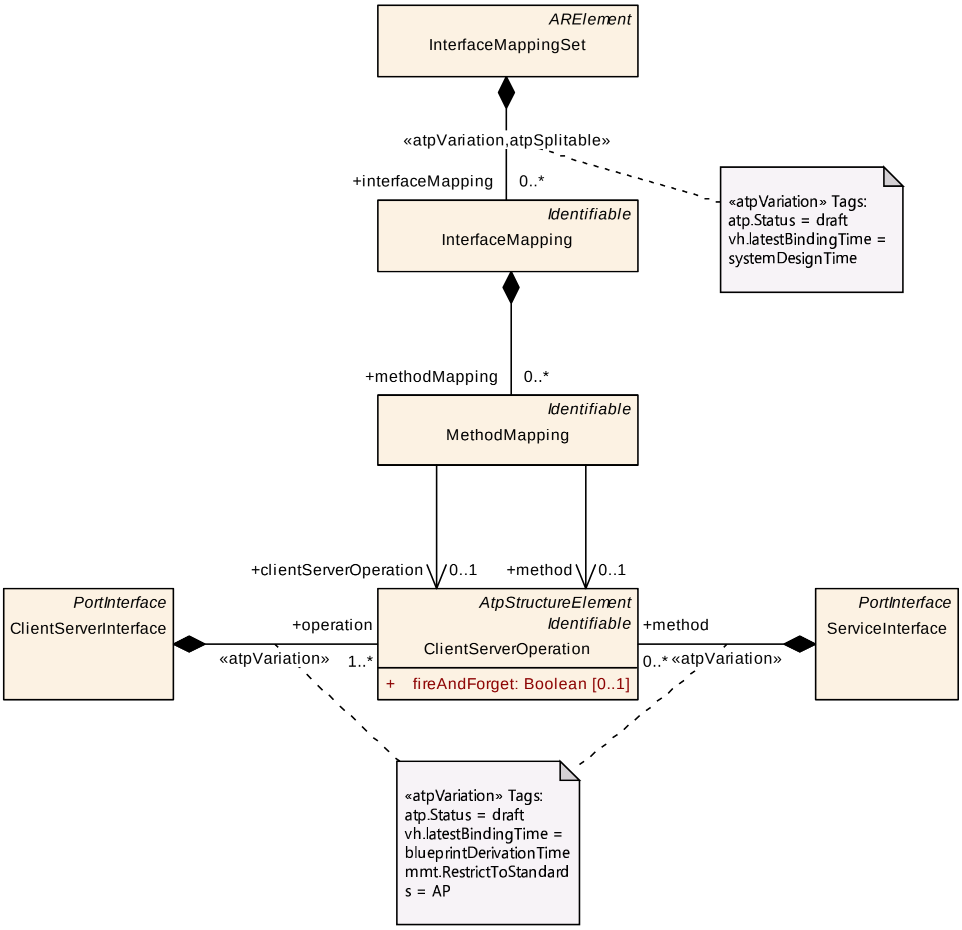


Figure 5.9: Mapping of a Method to a ClientServerOperation

Class	**MethodMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	Mapping of a ClientServerOperation that is located in a ClientServerInterface to a Method that is located in a ServiceInterface. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
clientServer Operation	ClientServerOperation	0..1	ref	Reference to a ClientSeverOperation that is located in a ClientSeverInterface. **Tags:** atp.Status=draft
method	ClientServerOperation	0..1	ref	Reference to a Method that is located in a Service Interface. **Tags:** atp.Status=draft

Table 5.23: MethodMapping

5.4.2 EventMapping

[TPS_MANI_03112]{DRAFT} **Mapping between an event and a dataElement** ⌈
The mapping between an event located in a ServiceInterface and a dataElement located in a SenderReceiverInterface is provided by the class EventMapping. ⌋*(RS_MANI_00026)*

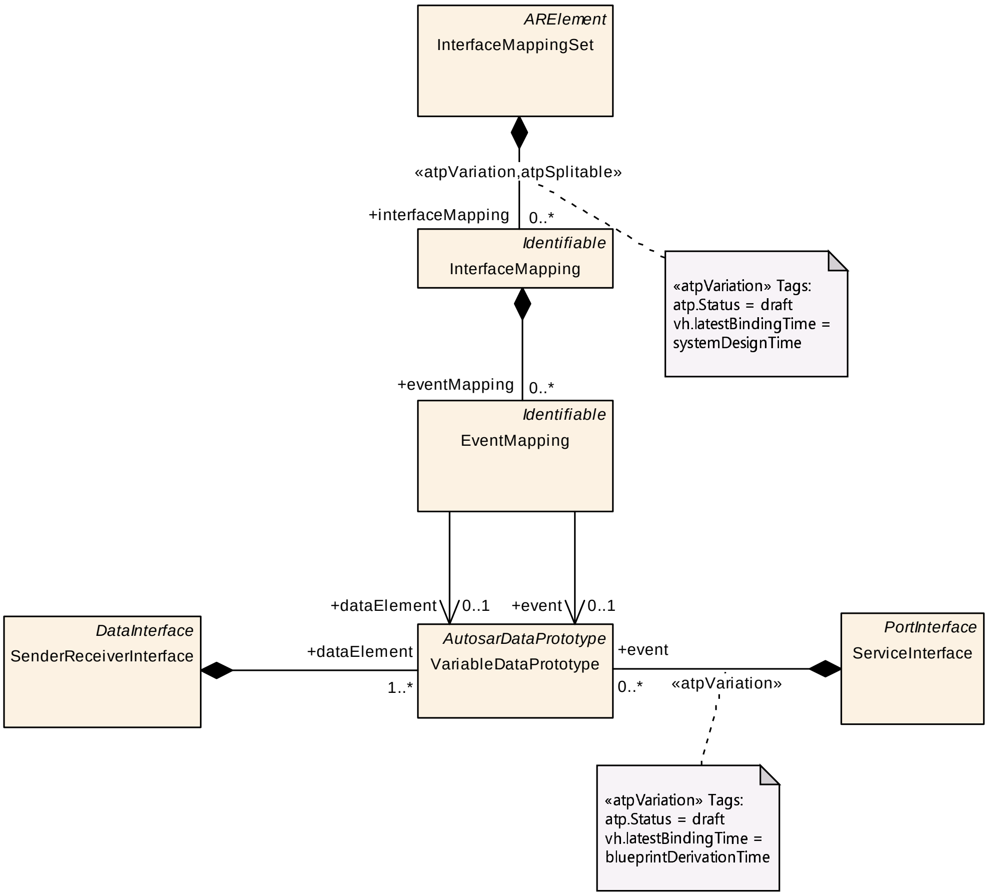


Figure 5.10: Mapping between an event and a dataElement

Class	**EventMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	Mapping of a VariableDataPrototype that is located in a SenderReceiverInterface to an Event that is located in a ServiceInterface. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataElement	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that is located in a SenderReceiverInterface. **Tags:** atp.Status=draft

▽

△

Class	EventMapping			
event	VariableDataPrototype	0..1	ref	Reference to an Event that is located in a Service Interface. **Tags:** atp.Status=draft

Table 5.24: EventMapping

5.4.3 FieldMapping

[TPS_MANI_03113]{DRAFT} **Mapping between a field and elements of Classic Platform PortInterfaces** ⌈ The mapping between a field located in a ServiceInterface and elements of Classic Platform PortInterfaces is provided by the class FieldMapping. The field notifier in the classic platform is represented by a dataElement that is located in a SenderReceiverInterface. The getter and setter methods in the classic platform are represented by operations that are located in a ClientServerInterface. ⌋*(RS_MANI_00026)*

[constr_3367]{DRAFT} **FieldMapping.notifierDataElement reference** ⌈ The FieldMapping shall only contain the notifierDataElement reference if the hasNotifier attribute in the referenced field is set to true. ⌋*()*

[constr_3368]{DRAFT} **FieldMapping.getterOperation reference** ⌈ The FieldMapping shall only contain the getterOperation reference if the hasGetter attribute in the referenced field is set to true. ⌋*()*

[constr_3369]{DRAFT} **FieldMapping.setterOperation reference** ⌈ The FieldMapping shall only contain the setterOperation reference if the hasSetter attribute in the referenced field is set to true. ⌋*()*

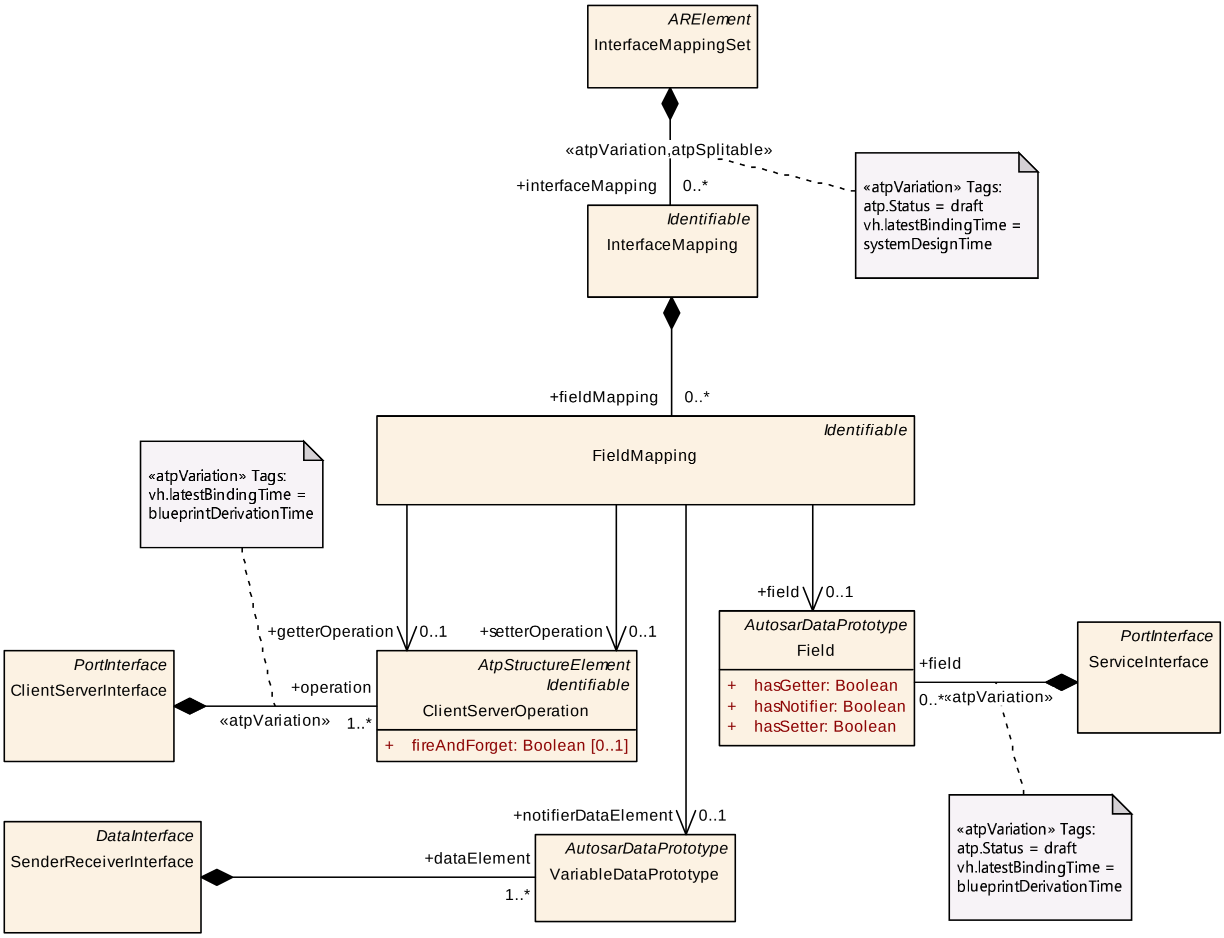


Figure 5.11: Mapping between a field and elements of Classic Platform PortInterfaces

Class	**FieldMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	Mapping of a Field that is located in a ServiceInterface to ClientServerOperations that represent the getter and setter methods and to a VariableDataPrototype that represents the notifier in the Field. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
field	Field	0..1	ref	Reference to a field that is located in a ServiceInterface. **Tags:** atp.Status=draft
getterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the getter Method in the Field. **Tags:** atp.Status=draft
notifierData Element	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that represents the notifier in the Field. **Tags:** atp.Status=draft

▽

△

Class	FieldMapping			
setterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the setter Method in the Field. **Tags:** atp.Status=draft

Table 5.25: FieldMapping

5.4.4 FireAndForgetMapping

In a fire and forget Message Exchange Pattern the consumer sends a message to a provider with no expectation of a response as described in chapter 3.4.4.1.

In Adaptive Autosar the fire and forget method is described with a `method` where the value of attribute `method.fireAndForget` is set to `true` as defined by [TPS_MANI_01064].

In classic Autosar a fire and forget method can not be described with a `ClientServerOperation` since a client-server call always has a response. Therefore a `VariableDataPrototype` is used if the fire and forget method contains input arguments. If the fire and forget method contains several input arguments then the `VariableDataPrototype` needs to be of type Structure that hosts one element for each argument of the fire and forget method. It is important that the order of elements in the Structure is the same as the order of `ArgumentDataPrototype`s within the `ClientServerOperation`.

This representation ensures that the SOME/IP serialization results in the same byte stream as in the Adaptive Platform where all `argument`s which have the `direction in` are serialized according to the order of the `ArgumentDataPrototype`s within the `ClientServerOperation`.

If the fire and forget method is without any parameters a `Trigger` is used to describe such a method in classic Autosar.

It is important that the SOME/IP MessageType is set to REQUEST_NO_RETURN if a fire and forget method is transmitted over SOME/IP.

[TPS_MANI_03115]{DRAFT} **Mapping between a fire and forget `method` and elements of Classic Platform `PortInterface`s** ⌈ The mapping between a `method` for which the value of attribute `method.fireAndForget` is set to `true` and elements of Classic Platform `PortInterface`s is provided by the class `FireAndForgetMapping`. If the fire and forget method is represented in the classic platform by a `VariableDataPrototype` then this `dataElement` is mapped to a `method` located in a `ServiceInterface`. If the fire and forget method is represented in the classic platform by a `Trigger` then this `trigger` is mapped to a `method` located in a `ServiceInterface`. ⌋*(RS_MANI_00026)*

[constr_3371]{DRAFT} **Mutually exclusive existence of FireAndForgetMapping.dataElement reference and FireAndForgetMapping.trigger reference**
⌈A FireAndForgetMapping shall never reference a dataElement and a trigger at the same time.⌋()

[constr_3376]{DRAFT} **FireAndForgetMapping shall reference only fire and forget methods** ⌈ A FireAndForgetMapping is only allowed to reference a ClientServerOperation in role method for which the value of attribute method.fireAndForget is set to true.⌋()

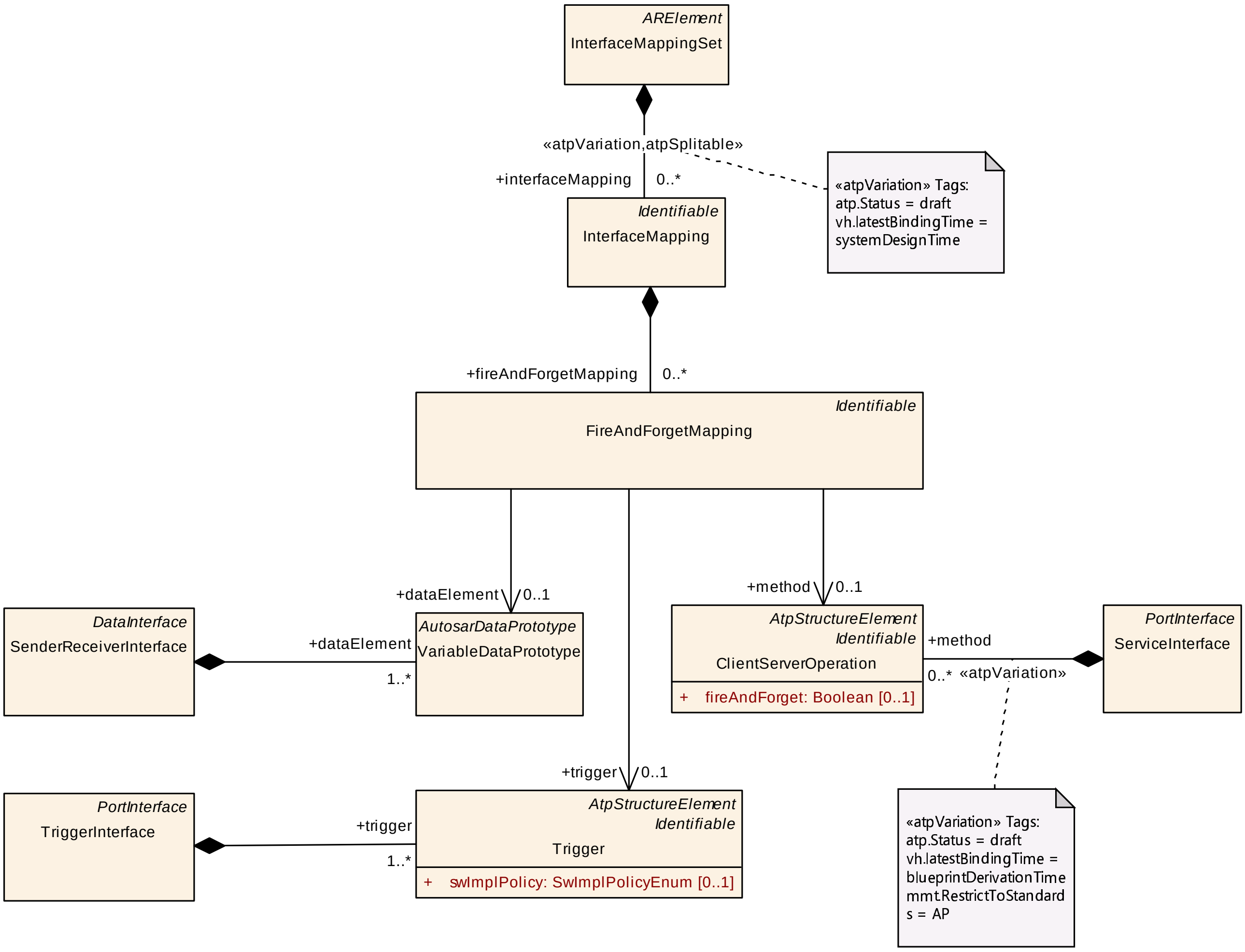


Figure 5.12: Mapping between a fire and forget method and elements of Classic Platform PortInterfaces

Class	**FireAndForgetMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign
Note	Mapping of a Fire&Forget Method that is located in a ServiceInterface to a VariableDataPrototype in a SenderReceiverInterface or to a Trigger in a TriggerInterface. **Tags:** atp.Status=draft

▽

△

Class	**FireAndForgetMapping**			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataElement	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that is located in a SenderReceiverInterface in case that the Fire&Forget Method is represented by this VariableDataPrototype. **Tags:** atp.Status=draft
method	ClientServerOperation	0..1	ref	Reference to a Fire&Forget Method that is located in a ServiceInterface. **Tags:** atp.Status=draft
trigger	Trigger	0..1	ref	Reference to a Trigger that is located in a TriggerInterface in case that the Fire&Forget Method is represented by this Trigger. **Tags:** atp.Status=draft

Table 5.26: FireAndForgetMapping

6 Machine Manifest

The Machine meta-class defines the entity on which one *Adaptive AUTOSAR Software Stack* is running with an operating system. The Machine may be physical or virtual.

Some aspects of the actual Machine are already available from the System Design (see chapter 5.2) at the MachineDesign. The information defined at the MachineDesign is available to the Machine as well since Machine has a reference to the MachineDesign in the role machineDesign (see figure 5.1).

The Machine is able to aggregate one or several Processors. And each Processor consists of one or several ProcessorCores.

Meta-class ProcessorCore provides attribute coreId that can be used e.g. in a bitmask to better control the utilization of processing resources.

[constr_1549]{DRAFT} **Value of ProcessorCore.coreId** ⌈ The value of ProcessorCore.coreId shall be unique in the context of the enclosing Processor. ⌋*()*

An overview of the Machine meta-class is sketched in Figure 6.1.

[TPS_MANI_03035]{DRAFT} **Content of the Machine configuration** ⌈ The purpose of the Machine is to provide machine specific configuration settings. ⌋ *(RS_MANI_00020, RS_MANI_00021, RS_MANI_00022, RS_MANI_00023)*

[TPS_MANI_01208]{DRAFT} **Definition of envionment variables in the scope of a Machine** ⌈ It is possible to define environment variables in the scope of the entire Machine.

For this purpose the aggregation of TagWithOptionalValue in the role Machine.environmentVariable exists.

The name of the environment variable shall be specified by means of the attribute TagWithOptionalValue.key, the value can be modeled by means of TagWithOptionalValue.value.

This encloses the ability to define environment variables with empty values. For this purpose, the attribute TagWithOptionalValue.value shall simply be omitted. ⌋ *(RS_MANI_00022, RS_MANI_00023)*

Please note that the aggregation Machine.environmentVariable has been defined with the stereotype ≪atpSplitable≫. The consequence of this modeling is that it is possible to contribute the definition of environment variables from **different sources**.

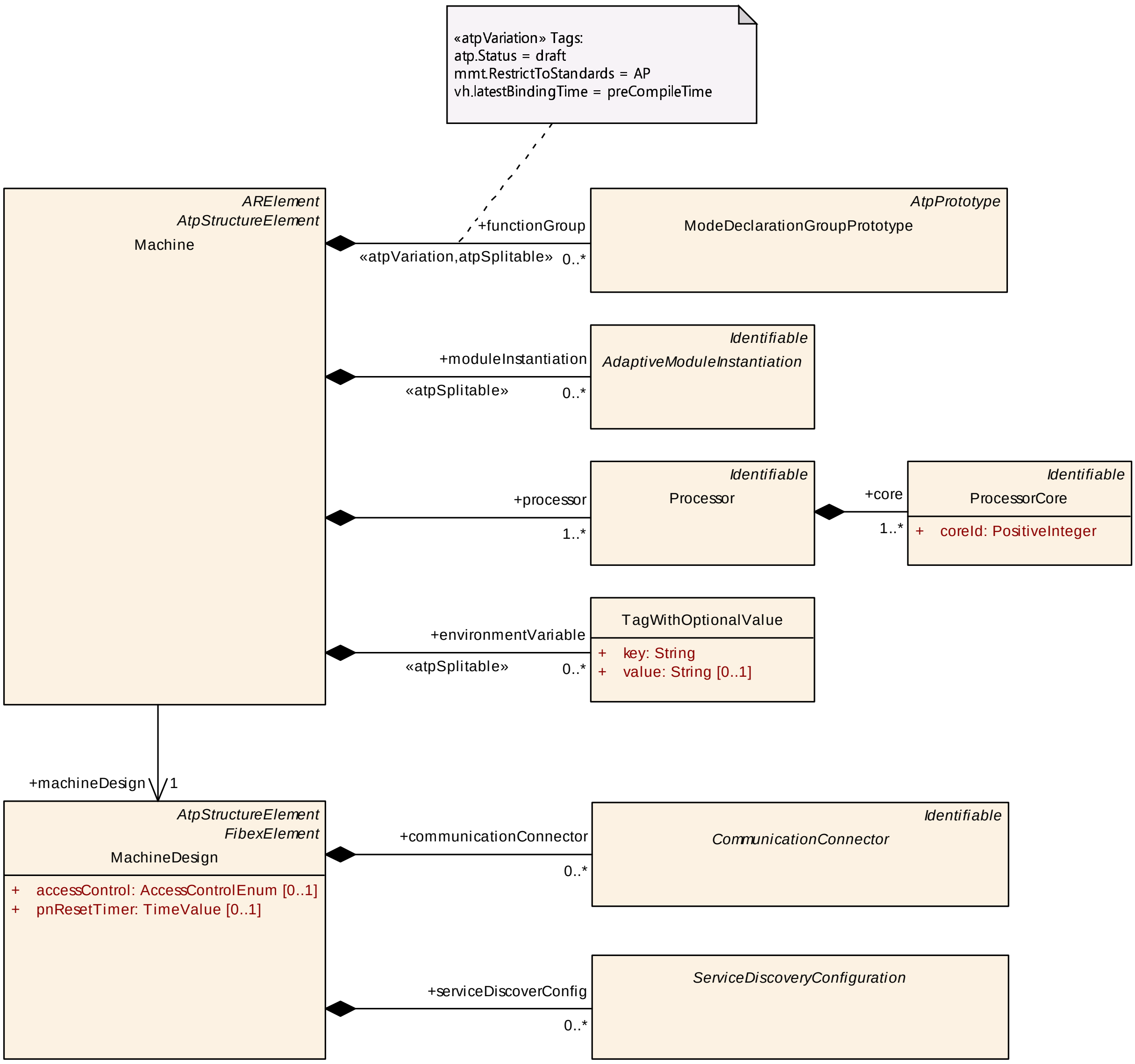


Figure 6.1: Overview about the content of the Machine configuration

Class	**Machine**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	Machine that represents an Adaptive Autosar Software Stack. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft atp.recommendedPackage=Machines			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
default Application Timeout	EnterExitTimeout	0..1	aggr	This aggration defines a default timeout in the context of a given Machine with respect to the launching and termination of applications. **Tags:** atp.Status=draft

▽

△

Class	**Machine**			
environment Variable	TagWithOptionalValue	*	aggr	This aggregation represents the collection of environment variables that shall be added to the environment defined on the level of the enclosing Machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=environmentVariable atp.Status=draft
functionGroup	ModeDeclarationGroup Prototype	*	aggr	This aggregation represents the collection of function groups of the enclosing Machine. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=preCompileTime
hwElement	HwElement	*	ref	This reference is used to describe the hardware resources of the machine. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
machineDesign	MachineDesign	1	ref	Reference to the MachineDesign this Machine is implementing. **Tags:** atp.Status=draft
module Instantiation	AdaptiveModule Instantiation	*	aggr	Configuration of Adaptive Autosar module instances that are running on the machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft
perState Timeout	PerStateTimeout	*	aggr	This aggregation represens the definition of per-state-timeouts in the context of the enclosing machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=perStateTimeout atp.Status=draft
processor	Processor	1..*	aggr	This represents the collection of processors owned by the enclosing machine. **Tags:** atp.Status=draft
secure Communication Deployment	SecureCommunication Deployment	*	aggr	Deployment of secure communication protocol configuration settings to crypto module entities. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel atp.Status=draft

Table 6.1: Machine

Class	**Processor**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This represents a processor for the execution of an AUTOSAR adaptive platform **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	**Processor**			
core	ProcessorCore	1..*	aggr	This represents the collection of cores owned by the enclosing processor. **Tags:** atp.Status=draft

Table 6.2: Processor

Class	**ProcessorCore**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class represents the ability to model a processor core for the execution of an AUTOSAR adaptive platform. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
coreId	PositiveInteger	1	attr	This attribute represents a numerical value assigned to the specific core. The value can be taken e.g. for use in a bitmask.

Table 6.3: ProcessorCore

6.1 Hardware Resources

[TPS_MANI_03065]{DRAFT} **Hardware resources of the machine** ⌈ With the Machine.hwElement reference it is possible to formally describe the hardware of the machine. ⌋*(RS_MANI_00020)*

The HwElement is the main describing element that is used for example to describe Processing units, memory, peripherals and sensors/actuators.

The HwCategory that is referenced by the HwElement defines the hardware type and the applicable attribute definitions are defined by HwAttributeDef. An attribute value can be assigned to HwAttributeDef by hwAttributeValue.

Predefined categories and corresponding attributes are described in the Ecu Resource Template [20].

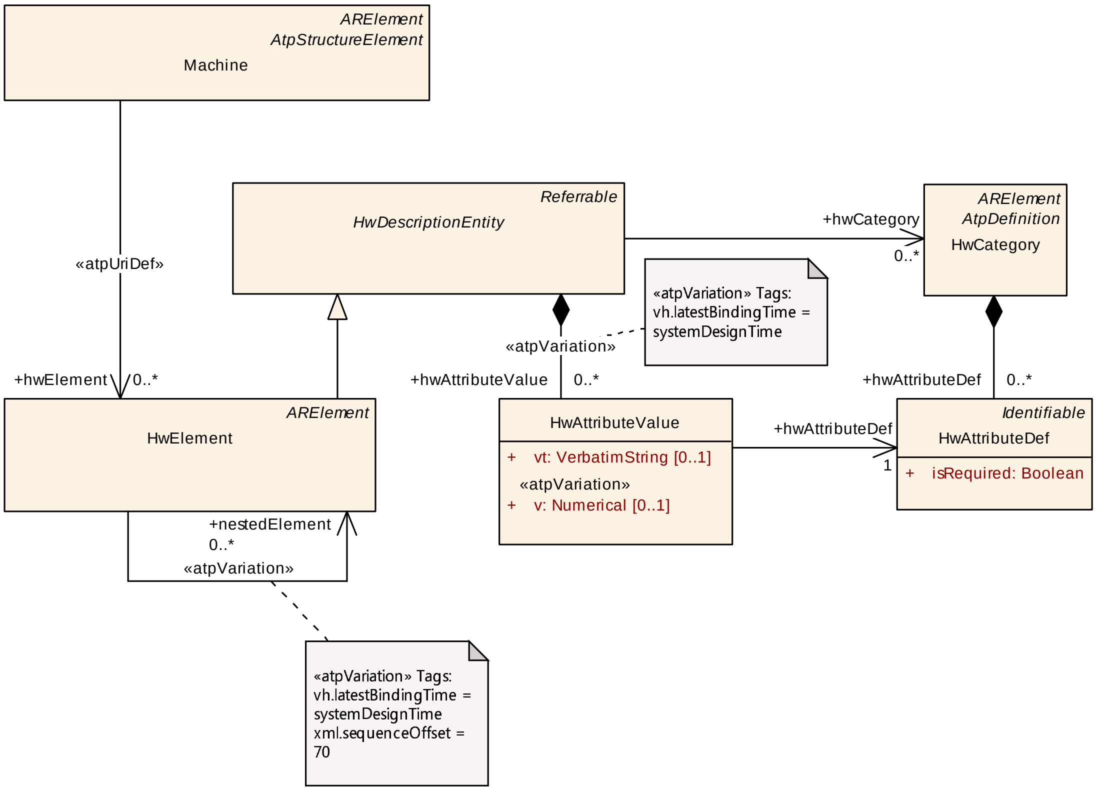


Figure 6.2: Description of hardware resources of the machine

Class	**HwElement**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This represents the ability to describe Hardware Elements on an instance level. The particular types of hardware are distinguished by the category. This category determines the applicable attributes. The possible categories and attributes are defined in HwCategory. **Tags:** atp.recommendedPackage=HwElements			
Base	*ARElement*, *ARObject*, *CollectableElement*, *HwDescriptionEntity*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
hwElement Connection	HwElementConnector	*	aggr	This represents one particular connection between two hardware elements. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=110
hwPinGroup	HwPinGroup	*	aggr	This aggregation is used to describe the connection facilities of a hardware element. Note that hardware element has no pins but only pingroups. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=90

▽

△

Class	**HwElement**			
nestedElement	HwElement	*	ref	This association is used to establish hierarchies of hw elements. Note that one particular HwElement can be target of this association only once. I.e. multiple instantiation of the same HwElement is not supported (at any hierarchy level). **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=70

Table 6.4: HwElement

Class	***HwDescriptionEntity*** (abstract)			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class represents the ability to describe a hardware entity.			
Base	*ARObject*, *Referrable*			
Subclasses	HwElement, HwPin, HwPinGroup, HwType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hwAttribute Value	HwAttributeValue	*	aggr	This aggregation represents a particular hardware attribute value. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=50
hwCategory	HwCategory	*	ref	One of the associations representing one particular category of the hardware entity. **Tags:** xml.sequenceOffset=30
hwType	HwType	0..1	ref	This association is used to assign an optional HwType which contains the common attribute values for all occurences of this HwDescriptionEntity. Note that HwTypes can not be redefined and therefore shall not have a hwType reference.

Table 6.5: HwDescriptionEntity

Class	**HwAttributeValue**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to assign a hardware attribute value. Note that v and vt are mutually exclusive.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
annotation	Annotation	0..1	aggr	Optional annotation that can be added to each Hw AttributeValue.
hwAttributeDef	HwAttributeDef	1	ref	This association represents the definition of the particular hardware attribute value.
v	Numerical	0..1	attr	This represents a numerical hardware attribute value. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime
vt	VerbatimString	0..1	attr	This represents a textual hardware attribute value.

Table 6.6: HwAttributeValue

Class	**HwCategory**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to declare hardware categories and its particular attributes. **Tags:** atp.recommendedPackage=HwCategorys			
Base	*ARElement*, *ARObject*, *AtpDefinition*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
hwAttributeDef	HwAttributeDef	*	aggr	This aggregation describes particular hardware attribute definition.

Table 6.7: HwCategory

Class	**HwAttributeDef**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to define a particular hardware attribute. The category of this element defines the type of the attributeValue. If the category is Enumeration the hw AttributeEnumerationLiterals specify the available literals.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
hwAttribute Literal	HwAttributeLiteralDef	*	aggr	The available EnumerationLiterals of the Enumeration definition. Only applicable if the category of the Hw AttributeDef equals Enumeration.
isRequired	Boolean	1	attr	This attribute specifies if the defined attribute value is required to be provided.
unit	Unit	0..1	ref	This association specifies the physical unit of the defined hardware attribute. This is optional due to the fact that there are textual attributes.

Table 6.8: HwAttributeDef

6.2 Function Groups

Function groups with function group states individually control groups of functionally coherent Application processes. The `Process` state may depend on a mode that is defined in the function group in case that the `StateDependentStartupConfig` refers to the function group state with the `functionGroupState` reference.

The usage of Function Groups is described in more detail in [17].

[TPS_MANI_03145]{DRAFT} **Description of a function group** ⌈ With the `functionGroup` aggregation it is possible to define a function group that has a `shortName` and a set of Modes (States) as `ModeDeclarationGroupPrototype` in the context of a `Machine`.

The `ModeDeclarationGroupPrototype` points to a reusable `ModeDeclarationGroup` in the role `type` that contains the different modes as `ModeDeclaration`s and a designated `initialMode`. ⌋*(RS_MANI_00041)*

[TPS_MANI_03194]{DRAFT} **Function Group State** ⌈ A function group state is described by a ModeDeclaration within a ModeDeclarationGroup that is referenced by a ModeDeclarationGroupPrototype aggregated as functionGroup by a Machine. The function group state is identified by its shortName. ⌋ *(RS_MANI_00041)*

[TPS_MANI_03195]{DRAFT} **Off state in Function Group** ⌈ Each functionGroup shall have an Off ModeDeclaration defined. This Off ModeDeclaration shall also be the initialMode of the functionGroup. ⌋ *(RS_MANI_00041)*

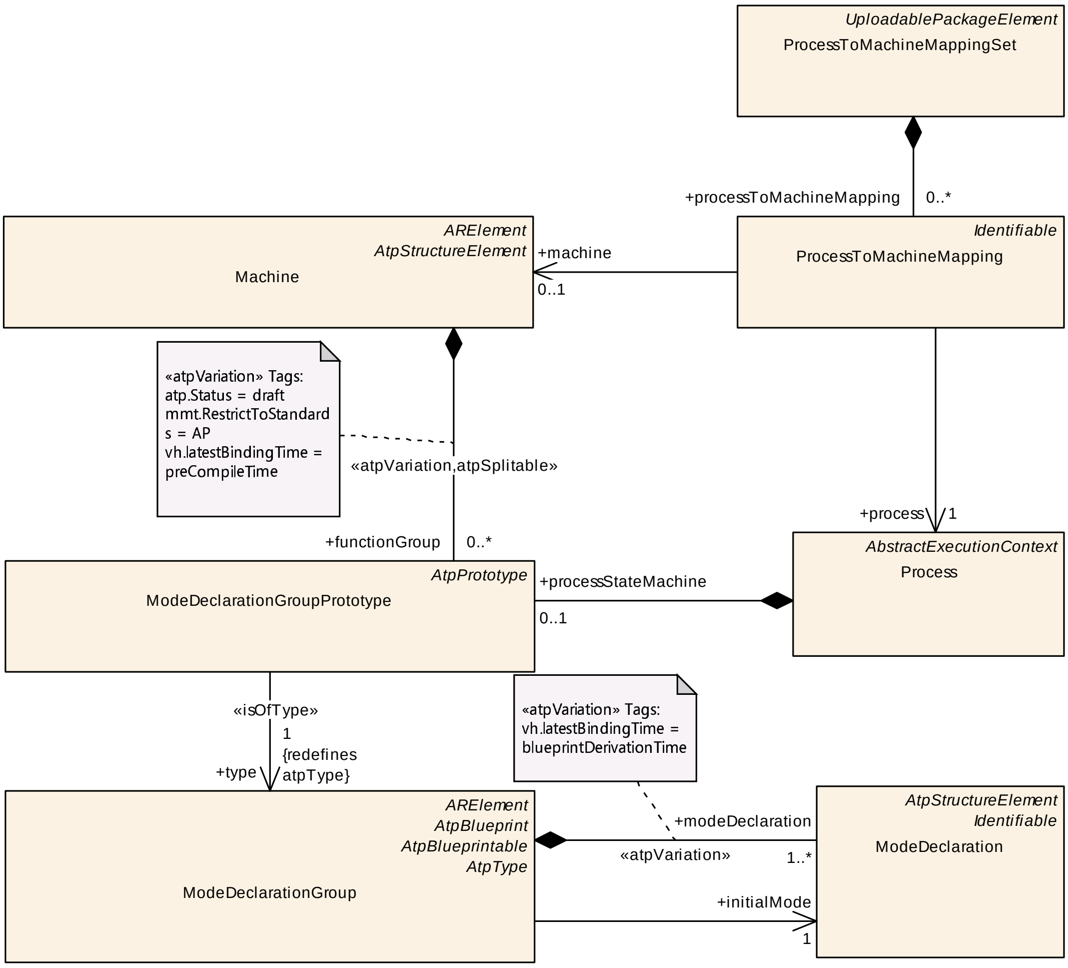


Figure 6.3: Configuration of Function Groups

Please note that it is required to define one specific functionGroup that takes the role of a "machine state". This functionGroup is required to have a dedicated shortName and it also is required to define a certain minimal, but extensible set of ModeDeclarations that also have standardized shortNames.

[constr_1687]{DRAFT} **Definition of machine state** ⌈ In the context of one Machine, exactly one functionGroup shall exist that has the shortName MachineState and that is typed be a ModeDeclarationGroup that defines at least the following list of ModeDeclarations with the shortNames

- Off,

- Startup,
- Shutdown, and
- Restart.

⌋()

Please note that the startup of a Process may depend on Modes that are defined in the context of a Machine. The StateDependentStartupConfig is described in chapter 7.2.

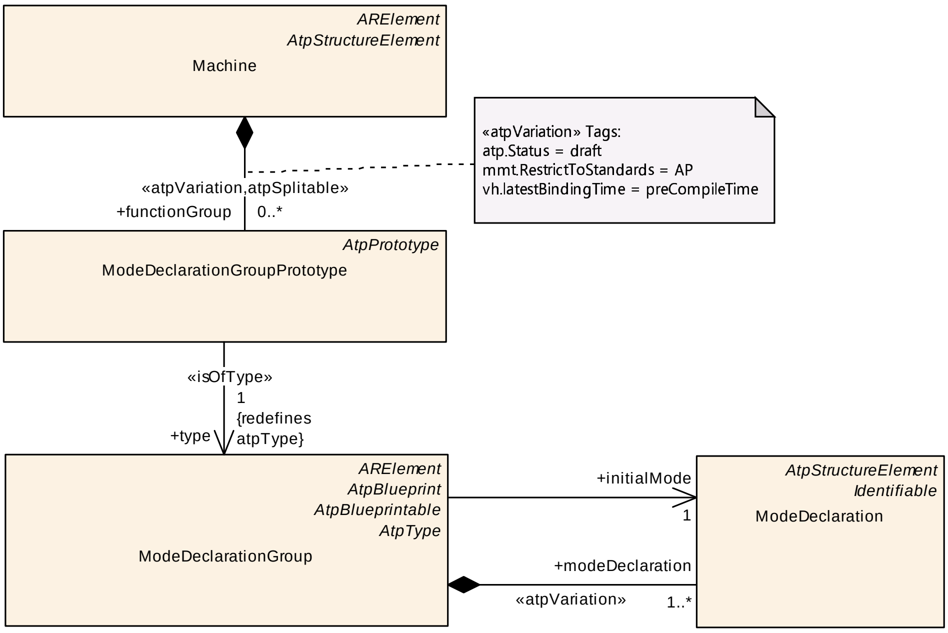


Figure 6.4: Configuration of Machine States

More details about the mandatory functionGroup that represents Machine States can be found in [17].

Class	**ModeDeclarationGroupPrototype**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context. **Tags:** atp.ManifestKind=ExecutionManifest,MachineManifest			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
type	ModeDeclarationGroup	1	tref	The "collection of ModeDeclarations" (= ModeDeclaration Group) supported by a component **Stereotypes:** isOfType

Table 6.9: ModeDeclarationGroupPrototype

Class	ModeDeclarationGroup			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.ManifestKind=ExecutionManifest,MachineManifest atp.recommendedPackage=ModeDeclarationGroups			
Base	*ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
initialMode	ModeDeclaration	1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
mode Declaration	ModeDeclaration	1..*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup

Table 6.10: ModeDeclarationGroup

Class	ModeDeclaration			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model. **Tags:** atp.ManifestKind=ExecutionManifest,MachineManifest			
Base	*ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 6.11: ModeDeclaration

6.3 State Timeouts

[TPS_MANI_03146]{DRAFT} **Configuration of timeouts for a selected machine state or function group state** ⌈ With the `PerStateTimeout` meta-class that is aggregated by the `Machine` in the role `perStateTimeout` it is possible to define `EnterExitTimeout`s for a selected machine state or function group state. The state for which the timeout is defined is specified by the `PerStateTimeout.state` reference.
⌋*()*

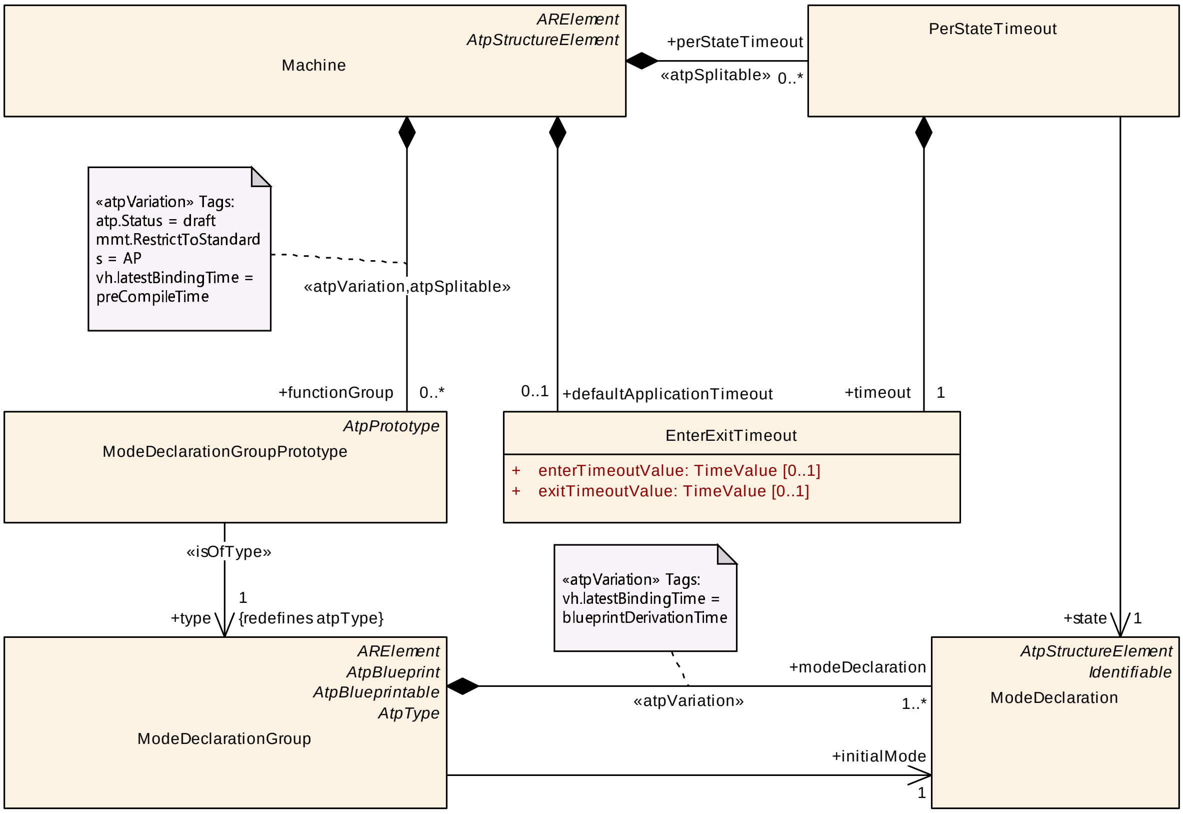


Figure 6.5: Configuration of timeouts for selected machine states and function group states

Class	**PerStateTimeout**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class represents the ability to specify a state-specific timeout. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
state	ModeDeclaration	1	ref	This reference represents the respective state for which the PerStateTimeout is defined. **Tags:** atp.Status=draft
timeout	EnterExitTimeout	1	aggr	This aggregation describes the timeout specification with respect to the referenced state. **Tags:** atp.Status=draft

Table 6.12: PerStateTimeout

The attribute `enterTimeoutValue` in the `EnterExitTimeout` meta-class defines the maximal time for start-up of all processes that are newly active in the referenced `state`.

The attribute `exitTimeoutValue` in the `EnterExitTimeout` meta-class defines the maximal time for termination of all processes that were active in the referenced `state` and are not assigned to a new `state`.

More details about the state timeouts are described in [17].

6.4 Process To Machine Mapping

6.4.1 General Modeling Approach

[TPS_MANI_03147]{DRAFT} **Mapping of a Process to a Machine** ⌈ The meta-class ProcessToMachineMapping provides the ability to map a Process to a Machine. ⌋*()*

[constr_1553]{DRAFT} **Restriction for ProcessToMachineMapping** ⌈ The following restrictions apply for the usage of ProcessToMachineMapping:

1. Each combination of Process and Machine shall only be referenced by one ProcessToMachineMapping in the role process or machine.
2. Each Process shall only be referenced by a single ProcessToMachineMapping in the role process.

⌋*()*

Please note that [constr_1553] does not imply that a given Machine shall only be referenced by a single ProcessToMachineMapping. It only says that one Process shall only be mapped once, to exactly one Machine.

[constr_5004]{DRAFT} **Mapping of a Process to a Machine is mandatory in the Execution Manifest** ⌈ Each Process shall be mapped by a ProcessToMachineMapping to one Machine. ⌋*()*

[constr_5004] means that a formal description of the assignment of a Process to a Machine shall be provided in the Execution Manifest, even though the Manifest will be uploaded to the Machine in combination with other artifacts to which the Manifest applies. The formal ProcessToMachineMapping was introduced because it is useful in the processing of the model in many cases.

Please note that according to the Autosar Methodology the Execution Manifest is created on the basis of an existing Machine Manifest and therefore the link to the Machine can always be created in the Execution Manifest.

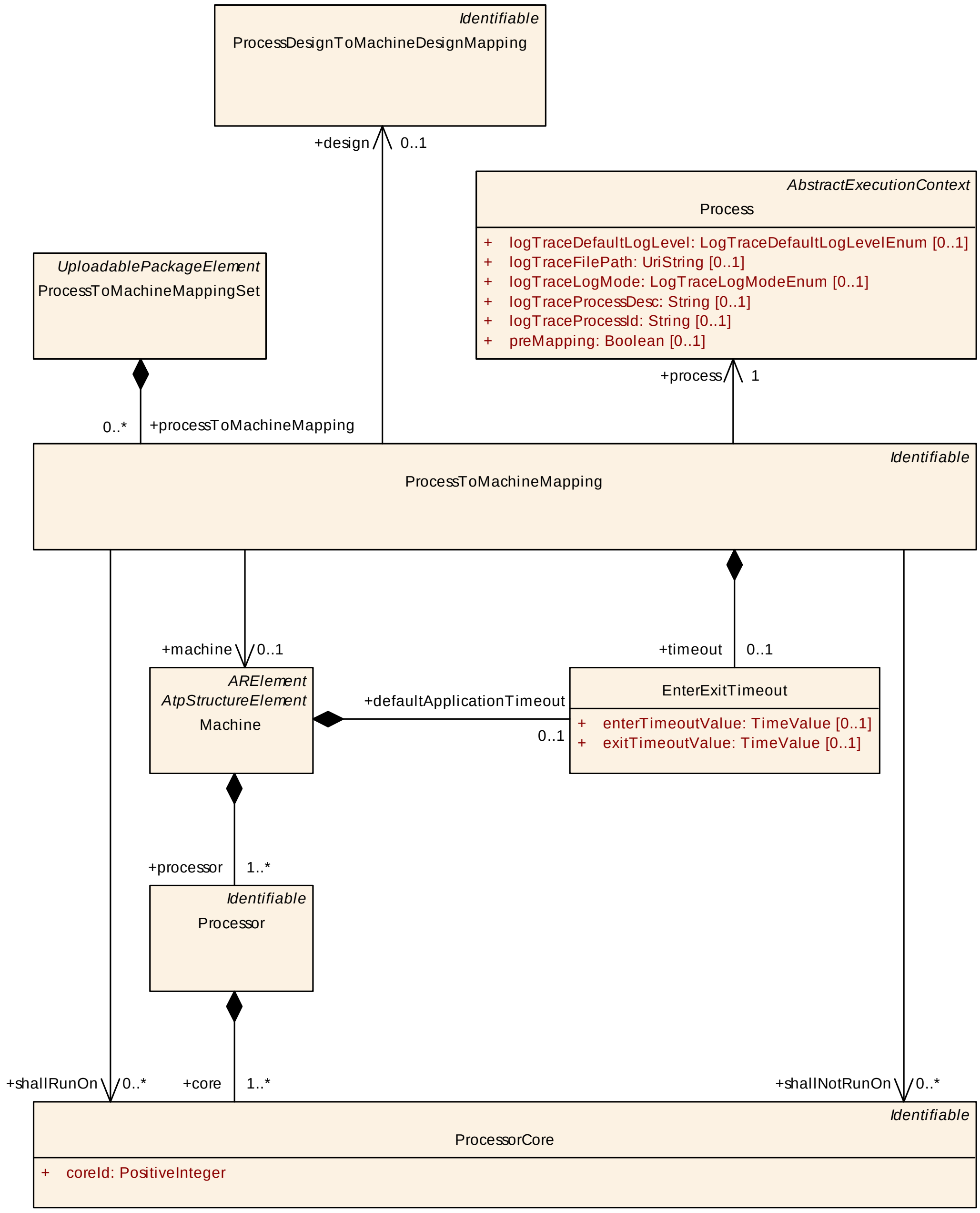


Figure 6.6: Mapping of a `Process` to a `Machine`

Class	**ProcessToMachineMappingSet**
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest
Note	This meta-class acts as a bucket for collecting ProcessToMachineMappings. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=ProcessToMachineMappings
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*

▽

△

Class	ProcessToMachineMappingSet			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
processToMachineMapping	ProcessToMachineMapping	*	aggr	This represents the collection of ProcessToMachineMappings of the enclosing ProcessToMachineMappingSet. **Tags:** atp.Status=draft

Table 6.13: ProcessToMachineMappingSet

Class	ProcessToMachineMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class has the ability to associate a Process with a Machine. This relation involves the definition of further properties, e.g. timeouts. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
design	ProcessDesignToMachineDesignMapping	0..1	ref	This reference represents the identification of the design-time representation for the ProcessToMachineMapping that owns the reference. **Tags:** atp.Status=draft
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the ProcessToMachineMapping. **Tags:** atp.Status=draft
nonOsModuleInstantiation	NonOsModuleInstantiation	0..1	ref	This supports the optional case that the process represents a platform module. **Tags:** atp.Status=draft
process	Process	1	ref	This reference identifies the Process in the context of the ProcessToMachineMapping. **Tags:** atp.Status=draft
shallNotRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall not be executing. **Tags:** atp.Status=draft
shallRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall be executing. **Tags:** atp.Status=draft
timeout	EnterExitTimeout	0..1	aggr	This aggregation can be used to specify the timeouts for launching and terminating the process. **Tags:** atp.Status=draft

Table 6.14: ProcessToMachineMapping

6.4.2 Core Affinity

[TPS_MANI_03148]{DRAFT} **Description of Core affinity** ⌈ The meta-class `ProcessToMachineMapping` provides the ability to restrict the assignment of processes to selected `ProcessorCore`s with the two references `shallRunOn` and `shallNotRunOn`. ⌋*()*

[constr_3393]{DRAFT} **Usage of shallRunOn and shallNotRunOn references** ⌈ The ProcessorCore that is referenced by a ProcessToMachineMapping in the role shallRunOn or shallNotRunOn shall be aggregated by the Machine that is referenced in the role machine by the same ProcessToMachineMapping. ⌋*()*

[constr_1676]{DRAFT} **Consistency of references shallRunOn and shallNotRunOn** ⌈ Within the context of one ProcessToMachineMapping, all ProcessorCores referenced in the role shallRunOn or shallNotRunOn shall be aggregated by the same Processor. ⌋*()*

If a model defines that a given Process shall run on a select set of ProcessorCores then there is hardly a use case to (in addition) also specify the opposite, i.e. that the Process shall not run on another set of ProcessorCores, and vice versa.

In other words, either there is a motivation to identify the ProcessorCores on which a Process is supposed to run or there is a motivation to do the exact opposite and specify the ProcessorCores where the Process is not supposed to run.

This conclusion provides the motivation for the existence of [constr_1677].

[constr_1677]{DRAFT} **Mutual exclusive existence of references shallRunOn and shallNotRunOn** ⌈ For any given ProcessToMachineMapping, either the reference in the role shallRunOn or the reference in the role shallNotRunOn may exist. ⌋*()*

6.4.3 Start-up and Termination Timeout

[TPS_MANI_03149]{DRAFT} **Definition of a start-up timeout for a Process** ⌈ The meta-class ProcessToMachineMapping provides the ability to define a start-up timeout for a Process with the attribute enterTimeoutValue that is available in the EnterExitTimeout meta-class that is aggregated by the ProcessToMachineMapping in the role timeout. ⌋*()*

[TPS_MANI_03150]{DRAFT} **Definition of a termination timeout for a Process** ⌈ The meta-class ProcessToMachineMapping provides the ability to define a termination timeout for a Process with the attribute exitTimeoutValue that is available in the EnterExitTimeout meta-class that is aggregated by the ProcessToMachineMapping in the role timeout. ⌋*()*

[TPS_MANI_03151]{DRAFT} **Default value for termination timeout** ⌈ The meta-class Machine provides the ability to define a default value for termination timeout of applications in the context of the Machine with the attribute exitTimeoutValue that is available in the EnterExitTimeout meta-class that is aggregated by the Machine in the role defaultApplicationTimeout. ⌋*()*

[constr_3394]{DRAFT} **Default value for start-up timeout on the `Machine` is not configurable** ⌈ The attribute `enterTimeoutValue` that is available in the `EnterExitTimeout` is not allowed to be used if the `EnterExitTimeout` is aggregated by the `Machine` in the role `defaultApplicationTimeout`. ⌋*()*

Class	**EnterExitTimeout**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class represents the ability to specify a pair of timeouts, one for entering, and one for exiting. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
enterTimeout Value	TimeValue	0..1	attr	This attribute represents the value of the enter timeout in seconds.
exitTimeout Value	TimeValue	0..1	attr	This attribute represents the value of the exit timeout in seconds.

Table 6.15: EnterExitTimeout

7 Execution Manifest

7.1 Overview

The purpose of the execution manifest is to provide information that is needed for the actual deployment of an application (formally modeled as an SwComponentType) onto the AUTOSAR adaptive platform.

One aspect of the deployment information is the provision of information that could in principle be provided as part of the application software code but which would make the application software code become very much bound to specific usage scenarios.

The general idea is to keep the application software code as independent as possible from the deployment scenario in order to increase the odds that the application software can be reused in different deployment scenarios.

In particular, the usage of PortPrototypes as a means to express communication with the "outside" of the application software allows for abstracting away the details (the concrete service instance identification) of the service configuration. As far as the model is concerned, the API between the application and the middleware is represented by the PortPrototype.

The application code does not use specific service instances but takes the PortPrototype as a symbolic replacement for this information. The specifics of this modeling aspect are described in section 9.

The top-level element of the Execution Manifest definition is the Process, in reference to the fact that the unit of deployment on the *AUTOSAR adaptive platform* is a binary that, at runtime, makes a POSIX process.

[TPS_MANI_01011]{DRAFT} **Connection between application design and application deployment** ⌈ The connection between the *application design* and the *application deployment* is implemented by means of a reference from meta-class Process to meta-class Executable in the role executable.

By modeling the reference in this direction it is possible to keep the design level independent of the deployment level and, at the same time, bind the deployment to a specific design. ⌋*(RS_MANI_00006)*

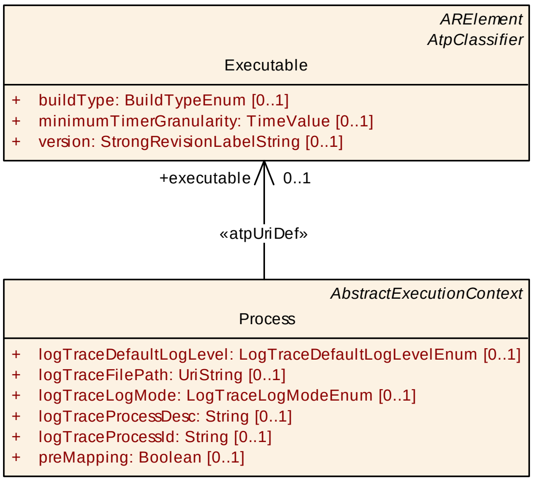


Figure 7.1: Relation of meta-classes `Executable` and `Process`

Class	**Process**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class provides information required to execute the referenced executable. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=Processes			
Base	*ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
design	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference. **Tags:** atp.Status=draft
deterministic Client	DeterministicClient	0..1	ref	This reference adds further execution characteristics for deterministic clients. **Tags:** atp.Status=draft
executable	Executable	0..1	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
logTraceDefault LogLevel	LogTraceDefaultLog LevelEnum	0..1	attr	This attribute allows to set the initial log reporting level for a logTraceProcessId (ApplicationId).
logTraceFile Path	UriString	0..1	attr	This attribute defines the destination file to which the logging information is passed.
logTraceLog Mode	LogTraceLogMode Enum	0..1	attr	This attribute defines the destination of log messages provided by the process.
logTrace ProcessDesc	String	0..1	attr	This attribute can be used to describe the logTrace ProcessId that is used in the log and trace message in more detail.
logTrace ProcessId	String	0..1	attr	This attribute identifies the process in the log and trace message (ApplicationId).
preMapping	Boolean	0..1	attr	This attribute describes whether the executable is preloaded into the memory.
processState Machine	ModeDeclarationGroup Prototype	0..1	aggr	Set of Process States that are defined for the process. **Tags:** atp.Status=draft

▽

△

Class	**Process**			
stateDependent StartupConfig	StateDependentStartup Config	*	aggr	Applicable startup configurations. **Tags:** atp.Status=draft

Table 7.1: Process

The preMapping approach of a Process is described in more detail in [SWS_EM_02109] in the SWS Execution Management [17].

Class	***AbstractExecutionContext*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class acts as a base class for entities that execute code on different levels, e.g. container, process, thread, fiber. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Subclasses	Process			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 7.2: AbstractExecutionContext

Please note that the meta-model, as depicted in Figure 7.1 supports the existence of two or more Processes that reference the same Executable.

This is an indication that the specific Executable is supposed to be executed in several instances (i.e. in the form of POSIX processes) on the same platform. Such a situation is sketched in Figure 7.2

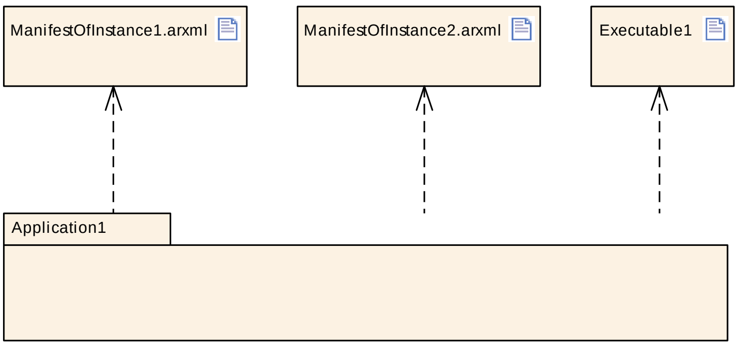


Figure 7.2: Example deployment where one Executable is bundled with two ARXML files that each contain the description of one Process

It is somehow likely that the startup conditions and startup parameters of different Processes may be different (in order to achieve a variation of the functionality of the Executable).

Therefore, it is necessary to allow for the definition of startup configurations on a per-Process-basis.

This aspect is described in section 7.2.

The supported process states that are defined in the Process.processStateMachine are described in more detail in [17] by [SWS_EM_01002], [SWS_EM_01003], [SWS_EM_01004], [SWS_EM_01005] and [SWS_EM_01006].

7.2 Startup Configuration

The configuration of startup behavior is an essential part of the execution manifest.

[TPS_MANI_01012]{DRAFT} **Formal modeling of application startup behavior** ⌈
The formal modeling of application startup behavior is implemented by means of the aggregation of meta-class StateDependentStartupConfig in the role Process.stateDependentStartupConfig. ⌋*(RS_MANI_00007)*

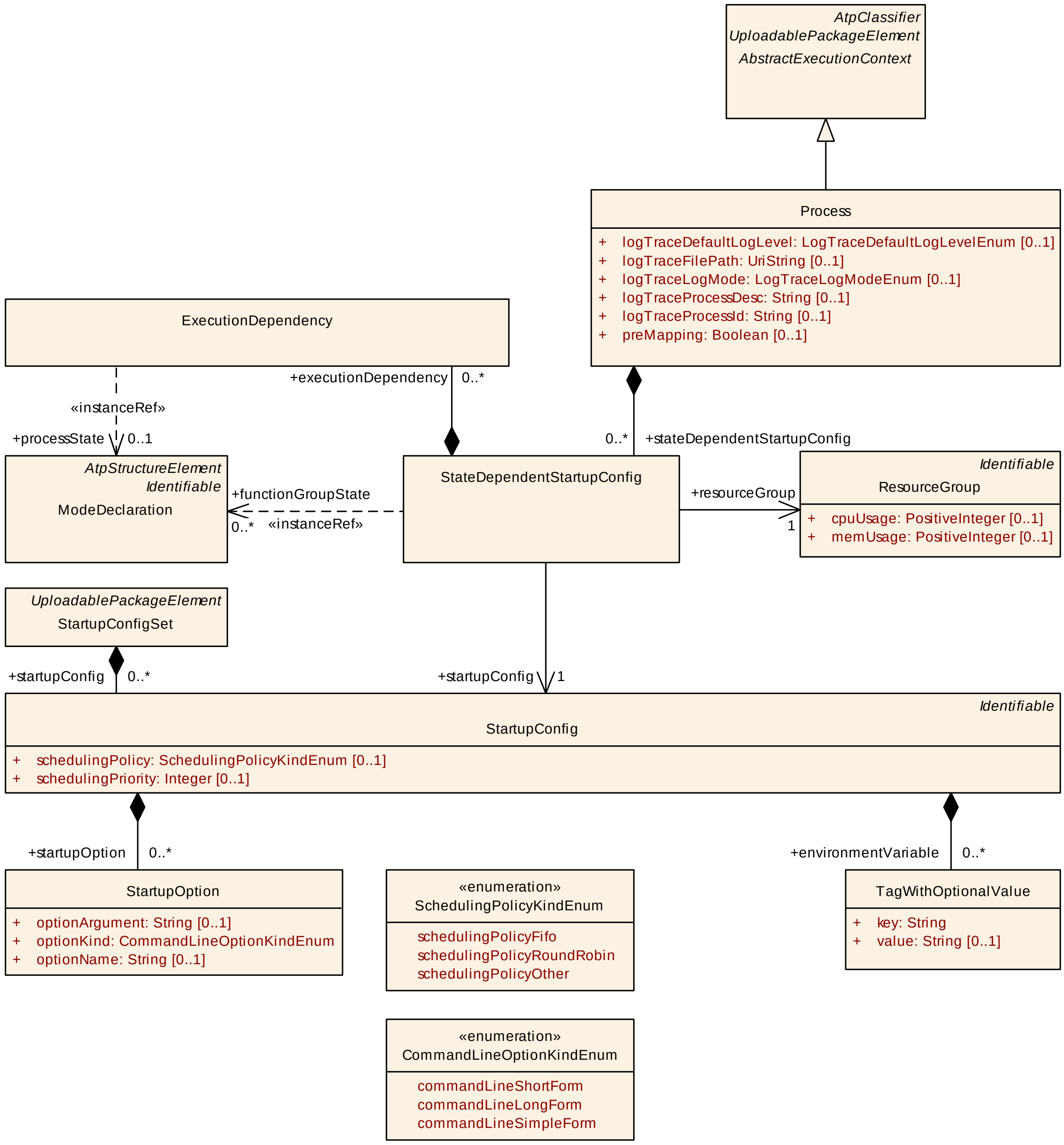


Figure 7.3: Content of a Process

7.2.1 State-dependent Startup Configuration

[TPS_MANI_01013]{DRAFT} **Semantics of meta-class StateDependentStartupConfig** ⌈ The purpose of meta-class StateDependentStartupConfig is to qualify the startup configuration represented by meta-class StartupConfig for specific ModeDeclarations.

In other words, the intention is to express that the StartupConfig is applicable if the state machines that control the startup are in the states represented by the ModeDeclaration referenced in the role StateDependentStartupConfig.functionGroupState. ⌋*(RS_MANI_00007)*

As a consequence of the reference from the StateDependentStartupConfig to ModeDeclaration the Execution Manifest is defined for a specific Machine to which the binary and the Manifest is deployed.

[constr_3423]{DRAFT} **StateDependentStartupConfig of a Process shall reference a functionGroupState** ⌈ Each StateDependentStartupConfig of a Process shall reference at least one ModeDeclaration in the role functionGroupState. ⌋*()*

However, the references to function group states within the context of one Process shall only refer to function group states **of the same function group**. This aspect is formalized by [constr_1688].

[constr_1688]{DRAFT} **StateDependentStartupConfig shall only refer to function group states of the same function group** ⌈ For all StateDependentStartupConfigs aggregated in the role Process.stateDependentStartupConfig, references in the role functionGroupState to ModeDeclaration shall only refer to ModeDeclarations aggregated by the same ModeDeclarationGroup in the context of the same ModeDeclarationGroupPrototype (that represents the actual function group). ⌋*()*

It is necessary to specify constraint [constr_3396] to regulate the number of StateDependentStartupConfigs that refer to the same ModeDeclaration in the context of one Process because the resulting startup configuration would be ambiguous.

[constr_3396]{DRAFT} **Number of Process.stateDependentStartupConfig that refer to the same functionGroupState** ⌈ Within the context of a given Process, no two StateDependentStartupConfigs shall refer to the same ModeDeclaration in the role functionGroupState. ⌋*()*

[TPS_MANI_01046]{DRAFT} **Semantics of StateDependentStartupConfig.functionGroupState** ⌈ The ModeDeclarations referenced in the role StateDependentStartupConfig.functionGroupState shall be considered in a way such that the StateDependentStartupConfig applies if **any** of the referenced ModeDeclarations is active.

In other words, the ModeDeclarations are or-ed for the determination of whether a StateDependentStartupConfig is applicable. ⌋*(RS_MANI_00007)*

[constr_3424]{DRAFT} **StateDependentStartupConfig shall never reference the functionGroupState Off** ⌈ A StateDependentStartupConfig shall never reference the ModeDeclaration that has the shortName Off in the role functionGroupState. Please note that the Off ModeDeclaration is a special state in a Function Group as defined by [TPS_MANI_03195]. ⌋*()*

[constr_1618]{DRAFT} **Ability to shut down** ⌈ In the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortName Shutdown. ⌋*()*

[constr_1619]{DRAFT} **Ability to restart** ⌈ In the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortName Restart. ⌋*()*

[TPS_MANI_01209]{DRAFT} **Definition of envionment variables in process scope** ⌈ It is possible to define environment variables in the scope of any given Process.

For this purpose the aggregation of TagWithOptionalValue in the role StartupConfig.environmentVariable exists.

The name of the environment variable shall be specified by means of the attribute TagWithOptionalValue.key, the value can be modeled by means of TagWithOptionalValue.value.

This encloses the ability to define environment variables with empty values. For this purpose, the attribute TagWithOptionalValue.value shall simply be omitted. ⌋ *(RS_MANI_00007)*

Class	**StateDependentStartupConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class defines the startup configuration for the process depending on a collection of machine states. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
execution Dependency	ExecutionDependency	*	aggr	This attribute defines that all processes that are referenced via the ExecutionDependency shall be launched and shall reach a certain ProcessState before the referencing process is started. **Tags:** atp.Status=draft
functionGroup State	ModeDeclaration	*	iref	This represent the applicable functionGroupMode. **Tags:** atp.Status=draft
resourceGroup	ResourceGroup	1	ref	Reference to an applicable resource group. **Tags:** atp.Status=draft
startupConfig	StartupConfig	1	ref	Reference to a reusable startup configuration with startup parameters. **Tags:** atp.Status=draft

Table 7.3: StateDependentStartupConfig

Class	StartupConfig			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class represents a reusable startup configuration for processes.. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
environment Variable	TagWithOptionalValue	*	aggr	This aggregation represents the collection of environment variables that shall be added to the respective Process's environment prior to launch. **Tags:** atp.Status=draft
scheduling Policy	SchedulingPolicyKind Enum	0..1	attr	This attribute represents the ability to define the scheduling policy for the initial thread of the application.
scheduling Priority	Integer	0..1	attr	This is the scheduling priority requested by the application itself.
startupOption	StartupOption	*	aggr	Applicable startup options **Tags:** atp.Status=draft

Table 7.4: StartupConfig

7.2.2 Scheduling

[TPS_MANI_01061]{DRAFT} **Requirements on scheduling** ⌈ The attributes StartupConfig.schedulingPolicy and StartupConfig.schedulingPriority make requirements on the scheduling of the process that is created out of launching the Executable, i.e. the "outer" scheduling.

The value of these attributes has no direct impact on the behavior of any "inner" scheduling of threads. ⌋*(RS_MANI_00007)*

[TPS_MANI_01188]{DRAFT} **Semantics of attribute schedulingPriority** ⌈ The value of attribute StartupConfig.schedulingPriority shall be interpreted such that the higher values represent a higher scheduling priority. ⌋*(RS_MANI_00007)*

[constr_1692]{DRAFT} **Value of schedulingPriority** ⌈ The value of attribute StartupConfig.schedulingPriority shall be set to a positive integer value. ⌋*()*

Enumeration	SchedulingPolicyKindEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest
Note	This meta-class provides a set of settings that allow for the specification of a scheduling policy. For a detailed description of the scheduling policies defined in the context of this meta-class, please refer to The Open Group Base Specifications Issue 7, IEEE Std 1003.1, 2013 Edition. **Tags:** atp.Status=draft
Literal	***Description***

▽

△

Enumeration	**SchedulingPolicyKindEnum**
schedulingPolicy Fifo	This attribute represents the setting for a FIFO scheduling policy. **Tags:** atp.EnumerationValue=0
schedulingPolicy Other	This attribute represents the setting for a custom scheduling policy. **Tags:** atp.EnumerationValue=2
schedulingPolicy RoundRobin	This attribute represents the setting for a round robin scheduling policy **Tags:** atp.EnumerationValue=1

Table 7.5: SchedulingPolicyKindEnum

7.2.3 Startup Options

[TPS_MANI_01014]{DRAFT} **Semantics of meta-class `StartupConfigSet`** ⌈ The existence of a mode-dependent startup procedure implies the existence of a number of `StartupConfig`s within a given project.

Meta-class `StartupConfigSet` is therefore used as some sort of bucket to collect a number of `StartupConfig`s. ⌋*(RS_MANI_00007)*

Class	**StartupConfigSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	Collection of reusable startup configurations for processes. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=StartupConfigSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
startupConfig	StartupConfig	*	aggr	Startup configuration that is contained in the Startup ConfigSet **Tags:** atp.Status=draft

Table 7.6: StartupConfigSet

A POSIX process is usually started by a parent process, on the *AUTOSAR adaptive platform* this boils down to the `Execution Manager`. It is possible to pass a number of command-line options along with the command to launch the process.

The command-line options are then evaluated and taken into account by the process internally. In principle, command-line options are just a collection of tokens separated by whitespaces.

In most cases, it is not enough to have single tokens passed to the program because then the semantics of an individual token would not be unambiguous.

Therefore, conventions have evolved how to structure the collection of command-line options for launching a program.

In particular, the conventions assume the definition of pairs of command-line tokens where one token takes the role of a qualifier and the other takes the role of the value of that qualifier (example: `-v 1.0` or `--version=1.0`).

Whether or not single tokens can have a meaning depends on the individual program. For the modeling of command-line options this means:

- The model shall be able to describe a pair of command tokens that form a higher semantics in the sense that one qualifies and the other provides a value for that qualifier (example: `-v 1.0` or `--version=1.0`).
- Single tokens may have a fully-specified semantics (example: `-h`).
- It shall also be possible to just pass arguments along without any further markup (example: `../docs/config.txt`)
- Arbitrary number of tokens may appear on the command line of a program

These conclusions, along with the intention of the *AUTOSAR adaptive platform* to model the command line in a detailed way (as opposed to one opaque string), lead to the modeling of meta-class `StartupOption`.

[TPS_MANI_01015]{DRAFT} **Semantics of meta-class `StartupOption`** ⌈ Each `StartupOption` represents a command-line parameter that may (depending on the value of `optionKind`, see [constr_1497] and [constr_1498]) consist of one or two token.

On top of that, it is possible to specify the convention for tokens to be arranged in order to make a valid command-line parameter. The convention is represented by attribute `optionKind`. ⌋*(RS_MANI_00007)*

[TPS_MANI_01059]{DRAFT} **Different values of `optionKind` within a `StartupConfig.startupOption`** ⌈ The attribute `optionKind` may have a different value for each `optionKind` within a given `StartupConfig`. ⌋*(RS_MANI_00007)*

A simpler form of the statement made by [TPS_MANI_01059] is to say that different styles of startup options can be mixed within the context of a `StartupConfig`.

Please note that the usage of the value `commandLineSimpleForm` for attribute `optionKind` implicitly supports the usage of so-called "indirect files" that contain a list of startup options in order to overcome limitations regarding the total length of startup options on the command line.

In this case the typical strategy is to define a lead-in token that signals the nature of the command-line option, e.g. `@config.txt`.

[constr_1497]{DRAFT} **Attribute `optionKind` set to `commandLineSimpleForm`** ⌈ For any `StartupOption` where attribute `optionKind` is set to `CommandLineOptionKindEnum.commandLineSimpleForm` the attribute `optionName` **shall not** and attribute `optionArgument` **shall** exist. ⌋*()*

[constr_1498]{DRAFT} **Attribute optionKind set to commandLineShortForm or commandLineLongForm** ⌈ For any StartupOption where attribute optionKind is set to value CommandLineOptionKindEnum.commandLineShortForm or CommandLineOptionKindEnum.commandLineLongForm the attribute optionName **shall** exist. ⌋*()*

Class	**StartupOption**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class represents a single startup option consisting of option name and an optional argument. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
optionArgument	String	0..1	attr	This attribute defines option value.
optionKind	CommandLineOptionKindEnum	1	attr	This attribute specifies the style how the command line options appear in the command line.
optionName	String	0..1	attr	This attribute defines option name.

Table 7.7: StartupOption

Enumeration	**CommandLineOptionKindEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest
Note	This enum defines the different styles how the command line option appears in the command line. **Tags:** atp.Status=draft
Literal	***Description***
commandLineLongForm	Long form of command line option. Example: –version=1.0 –help **Tags:** atp.EnumerationValue=1
commandLineShortForm	Short form of command line option. Example: -v 1.0 -h **Tags:** atp.EnumerationValue=0
commandLineSimpleForm	In this case the command line option does not have any formal structure. Just the value is passed to the program. **Tags:** atp.EnumerationValue=2

Table 7.8: CommandLineOptionKindEnum

7.2.4 Resources

Meta-class StateDependentStartupConfig also supports the specification of a relation to a resource group.

[TPS_MANI_01017]{DRAFT} **Relation of startup configuration to resource group**
⌈ The modeling of a resource group is possible by means of meta-class ResourceGroup in the OsModuleInstantiation of the Machine and the assignment of a Process to a ResourceGroup is supported by the association from StateDependentStartupConfig to ResourceGroup in the role resourceGroup. ⌋ *(RS_MANI_00007)*

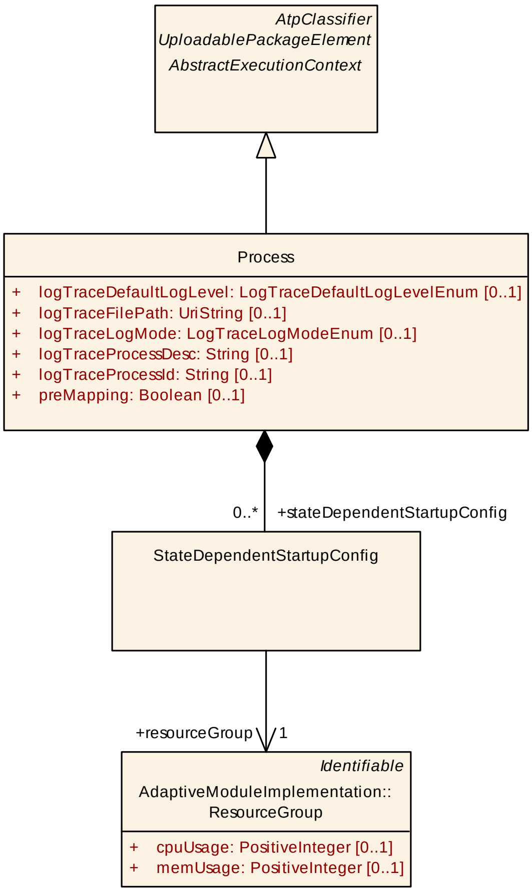


Figure 7.4: Modeling of how Process relates to ResourceGroup

[constr_3413]{DRAFT} **StateDependentStartupConfig of a Process is mapped to exactly one ResourceGroup** ⌈ Each StateDependentStartupConfig of a Process shall be assigned to exactly one ResourceGroup that is defined in the Machine Manifest. ⌋ *()*

7.2.5 Execution Dependency

The modeling of an execution dependency makes two Processes become associated to each other by means of the definition of an ExecutionDependency.

But since the reference that defines the execution dependency is modeled as an ≪instanceRef≫ the referenced Process needs to be extracted from the context references in the ≪instanceRef≫.

Once the two Processes are identified it is necessary for the validity of the startup dependency that they refer to the identical function group.

[TPS_MANI_01041]{DRAFT} **Startup configuration supports the definition of a launch sequence dependency** ⌈ The modeling of startup configuration also supports the definition of a launch sequence dependency, formalized by the meta-class ExecutionDependency that is aggregated by StateDependentStartupConfig in the role executionDependency.

The ExecutionDependency allows to define a dependency to a process that needs to be in a specific process state before the process that aggregates the ExecutionDependency via StateDependentStartupConfig is launched. ⌋ *(RS_MANI_00007)*

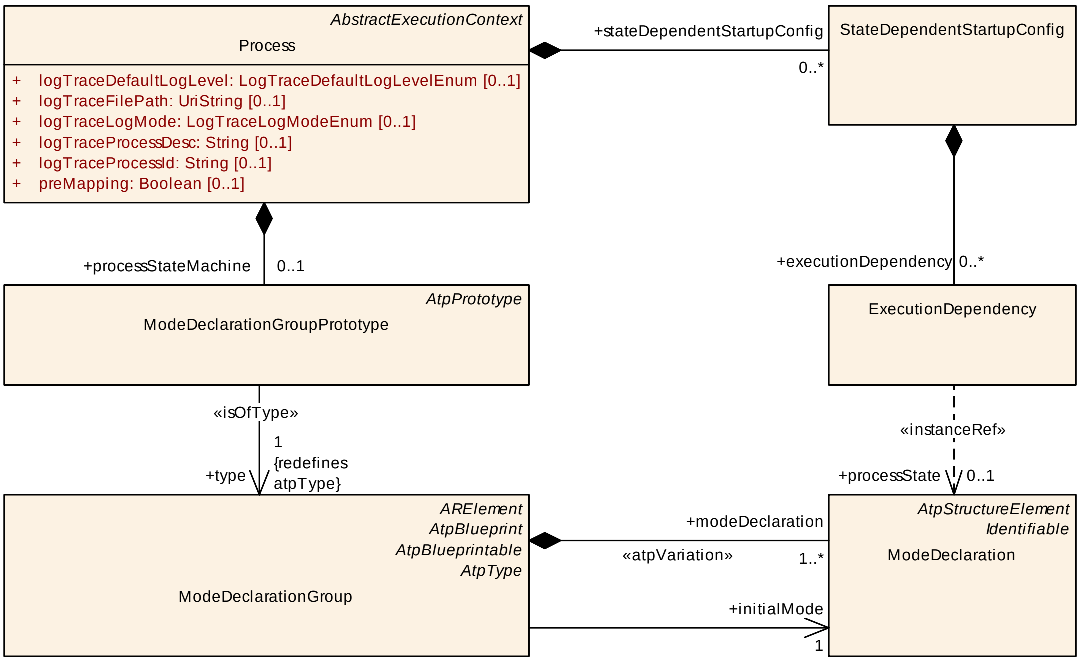


Figure 7.5: Modeling of how Process relates to ModeDeclaration owned by another Process

[constr_1689]{DRAFT} **Modeling of a startup dependency between different Processes** ⌈ The existence of attribute Process.stateDependentStartupConfig.executionDependency is only valid if the owner of the stateDependentStartupConfig.executionDependency (in other words: the referencing Process) and the owner of the ModeDeclarationGroupPrototype referenced in the role contextModeDeclarationGroupPrototype within the reference stateDependentStartupConfig.executionDependency.processState (i.e. the referenced Process) refer to the identical function group state formalized as ModeDeclaration. ⌋*()*

Figure 7.6 provides an exemplary explanation of [constr_1689]. In this example, Process “B” (the referencing Process as of [constr_1689]) defines an executionDependency to Process “A”.

This executionDependency is only valid if both Process “A” and Process “B” aggregate a StateDependentStartupConfig that refers to the same function group state “MD” within function group “FG”.

Process “A“ can be found by following the ExecutionDependency (specifically the contextModeDeclarationGroupPrototype) and the ≪instanceRef≫ that goes from the ExecutionDependency to the Process State “PS”.

The **owner** of “PS” is Process “B”, and if “B” refers to function group state “MD” within function group “FG” and if “A” refers to function group state“MD” within “FG” then the constraint [constr_1689] is fulfilled.

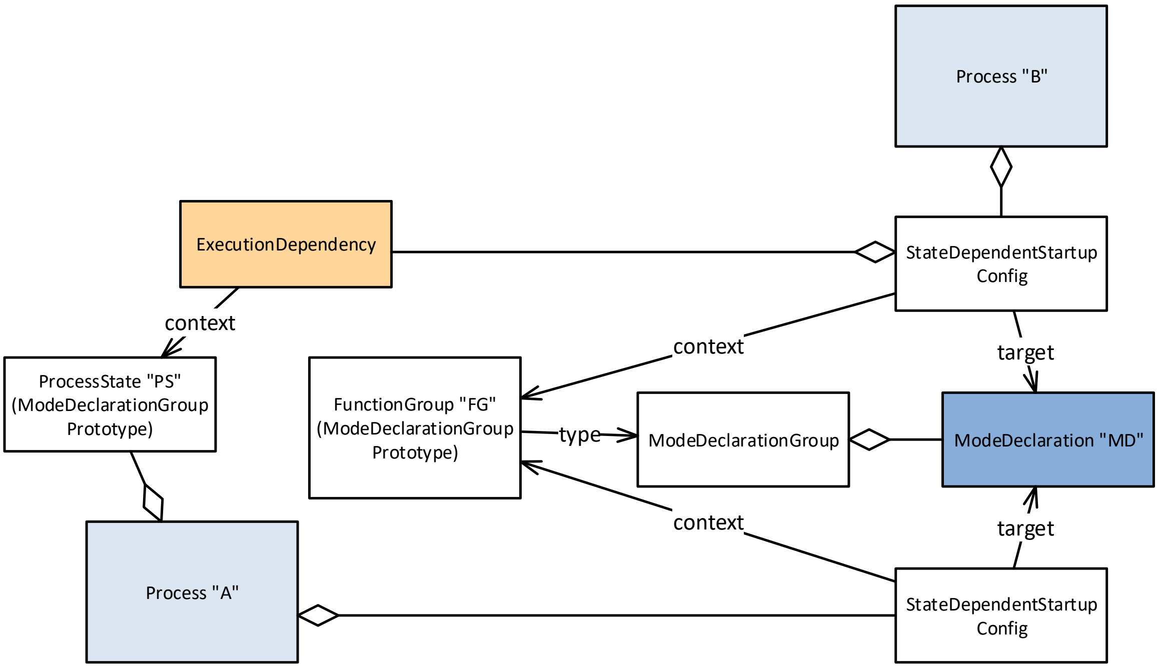


Figure 7.6: Explanation of dependencies from one Process to another

Class	**ExecutionDependency**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This element defines a ProcessState in which a dependent process needs to be before the process that aggregates the ExecutionDependency element can be started. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
processState	ModeDeclaration	0..1	iref	This represent the applicable modeDeclaration that represents an ProcessState. **Tags:** atp.Status=draft

Table 7.9: ExecutionDependency

[constr_1606]{DRAFT} **Processes with mutual ExecutionDependencys** ⌈ A Process.stateDependentStartupConfig.executionDependency shall not refer to any ModeDeclaration owned by a second Process that in turn refers via stateDependentStartupConfig.executionDependency to any ModeDeclaration owned by the first Process. ⌋*()*

7.2.6 Assignment of Processes to Function Group states

There are use cases where starting and terminating of individual groups of processes is necessary. This is supported in AUTOSAR by function groups that group processes together.

A function group may have a number of function group states, e.g. Running, Idle, Terminating. The StateDependentStartupConfig of a Process can be assigned to a function group state and the start-up of the Process will then depend on this assignment.

The modeling of function groups and their function group states is described in section 6.2 in more detail. The usage of Function Groups is described in more detail in [17].

[TPS_MANI_03152]{DRAFT} **Assignment of a StateDependentStartupConfig to a function group state** ⌈ The StateDependentStartupConfig is assigned to a function group state with the functionGroupState reference. ⌋*(RS_MANI_00041)*

7.3 Deterministic Client

As already explained in section 3.16.1, there is a use case to support the concept of the so-called Deterministic Client on the *AUTOSAR adaptive platform*. The conceptual background of Deterministic Client is explained in the SWS Execution Management [17].

The support for this concept consists of two aspects. The *design aspect* has already been explained in section 3.16.1 while the *deployment aspect* is discussed in this chapter.

[TPS_MANI_01203]{DRAFT} **Semantics of DeterministicClient** ⌈ The existence of reference Process.deterministicClient means that the enclosing Process implements the concept of a Deterministic Client.

Further information for the configuration of the Deterministic Client can be obtained from the ProcessDesign referenced in the role Process.design. ⌋*(RS_MANI_00050)*

The details of the support for Deterministic Client are visualized in Figure 7.7.

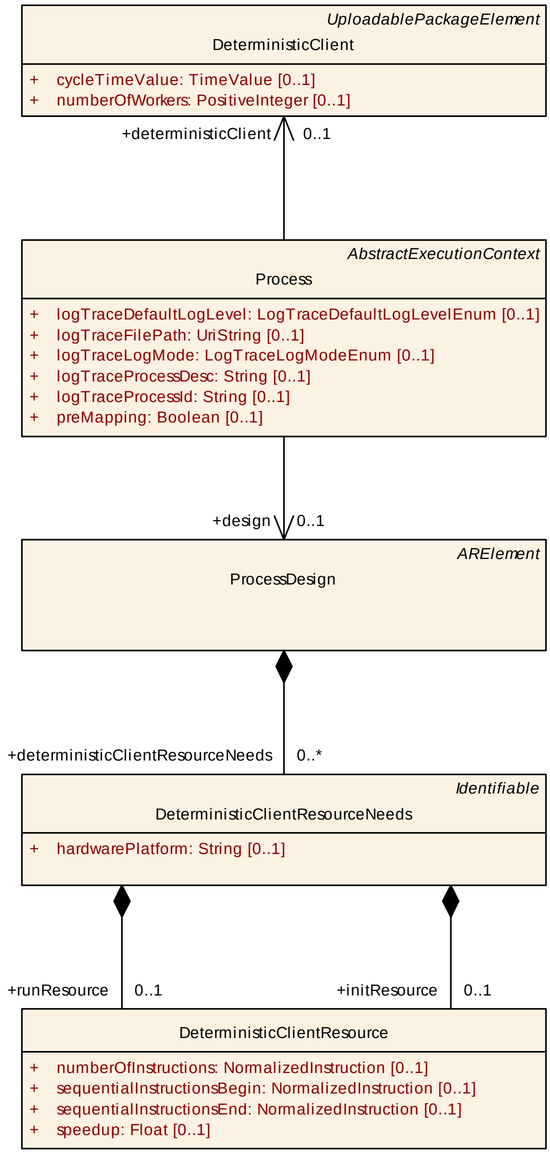


Figure 7.7: Modeling of suport for `Deterministic Client` in the deployment

Class	**DeterministicClient**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	The meta-class DeterministicClient provides the ability to support the deterministic execution of one or more processes with specific configuration parameters for DeterministicClient library functions. **Tags:** atp.Status=draft atp.recommendedPackage=DeterministicClients			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
cycleTimeValue	TimeValue	0..1	attr	This attribute represents the cycle time for execution of a DeterministicClient activation cycle.

▽

△

Class	**DeterministicClient**			
numberOf Workers	PositiveInteger	0..1	attr	Number of independent workers that process data-sets. Size of the worker pool shall be decided based on availability of resources like processor cores or memory.

Table 7.10: DeterministicClient

8 Platform Module Development

The configuration settings for individual Adaptive Autosar modules are covered by specializations of the abstract class `AdaptiveModuleInstantiation`.

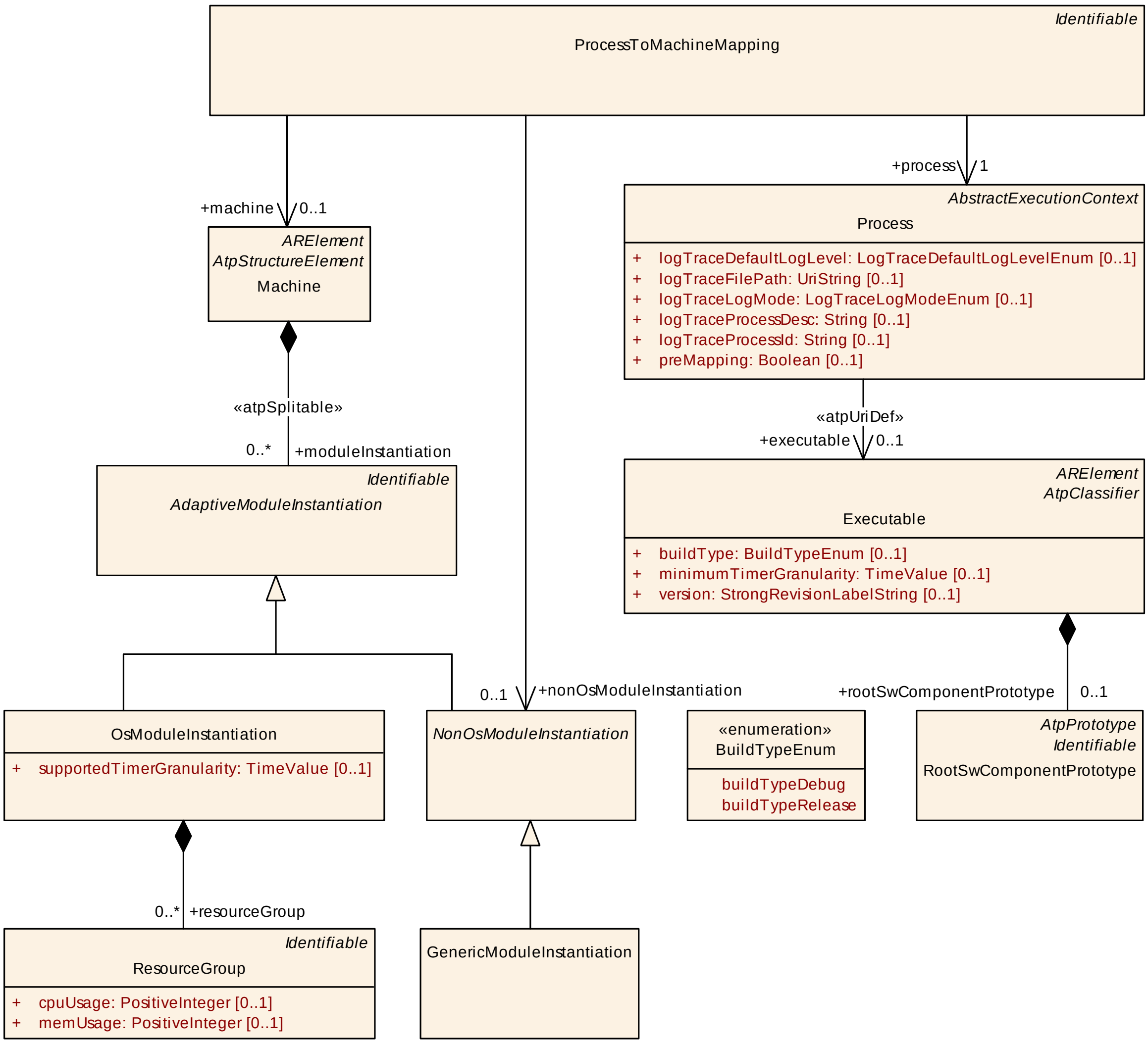


Figure 8.1: Adaptive Autosar Module Configuration

Class	***AdaptiveModuleInstantiation*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module instance on a specific machine. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	*NonOsModuleInstantiation*, OsModuleInstantiation

▽

△

Class	***AdaptiveModuleInstantiation*** (abstract)			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.1: AdaptiveModuleInstantiation

Each Adaptive Autosar module other than OS can be assigned to a Process with the ProcessToMachineMapping.

[constr_1490]{DRAFT} **Allowed value of category for reference ProcessToMachineMapping.process.executable** ⌈ The value of category of an Executable referenced in the role ProcessToMachineMapping.process.executable shall **only** be set to PLATFORM_LEVEL (see [constr_1605]). ⌋*()*

The meta-class GenericModuleInstantiation can be used to define configuration settings of generic modules and modules that are not standardized by AUTOSAR. Different modules are distinguishable by the category attribute.

Please note that both elements are Identifiable and therefore are able to describe special data (sdg), by which means it is possible to define generic custom settings that are not represented by the standard model. For more information, please refer to the AUTOSAR Generic Structure Template [6].

[TPS_MANI_03096]{DRAFT} **Machine-specific configuration settings for a generic module** ⌈ The Machine-specific configuration settings for a generic module are collected in GenericModuleInstantiation where the value of attribute category value denotes the module. ⌋*(RS_MANI_00023)*

Class	**GenericModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the generic module configuration on a specific machine. Different modules are distinguishable by the category attribute. This element can also be used to describe modules that are not standardized by AUTOSAR. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModule Instantiation*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.2: GenericModuleInstantiation

8.1 OS Module configuration

[TPS_MANI_03098]{DRAFT} **Machine-specific configuration settings for the OS module** ⌈ The Machine-specific configuration settings for the OS module are collected in OsModuleInstantiation. ⌋*(RS_MANI_00023)*

Class	OsModuleInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the OS configuration on a specific machine. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject, AdaptiveModuleInstantiation, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
resourceGroup	ResourceGroup	*	aggr	This represents the collection of ResourceGroups owned by the enclosing OsModuleImplementation. **Tags:** atp.Status=draft
supportedTimer Granularity	TimeValue	0..1	attr	This attribute describes the supported timer granularity (TimeValue of one tick). **Tags:** atp.Status=draft

Table 8.3: OsModuleInstantiation

AUTOSAR supports the configuration of ResourceGroups in the OsModuleInstantiation of the Machine that correspond for example to cgroups (aka control groups) in Linux. ResourceGroups provide a mechanism to manage system resources by partitioning constraints like cpuUsage and memUsage into groups that limit the resource usage for a collection of processes (see also [TPS_MANI_01017]).

[constr_1661]{DRAFT} **Multiplicity of OsModuleInstantiation.resourceGroup** ⌈ Any given OsModuleInstantiation shall always define at least one resourceGroup. ⌋*()*

The rationale for [constr_1661] is that the StateDependentStartupConfig requires a reference to a ResourceGroup.

Class	ResourceGroup			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class represents a resource group that limits the resource usage of a collection of processes. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
cpuUsage	PositiveInteger	0..1	attr	CPU resource limit in percentage of the total CPU capacity on the machine.
memUsage	PositiveInteger	0..1	attr	Memory limit in bytes.

Table 8.4: ResourceGroup

[constr_3412]{DRAFT} **OsModuleInstantiation shall have at least one ResourceGroup** ⌈ An OsModuleInstantiation in the Machine shall own at least one ResourceGroup. ⌋*()*

8.2 Persistency Deployment

8.2.1 Overview

This chapter explains the part of the support for persistent storage in terms of mapping of concrete storage models to the corresponding parts of the application software.

[TPS_MANI_01205]{DRAFT} **Semantics of meta-class** `PersistencyDeployment` ⌈Abstract meta-class `PersistencyDeployment` provides shared attributes to more specific specializations. ⌋*(RS_MANI_00027)*

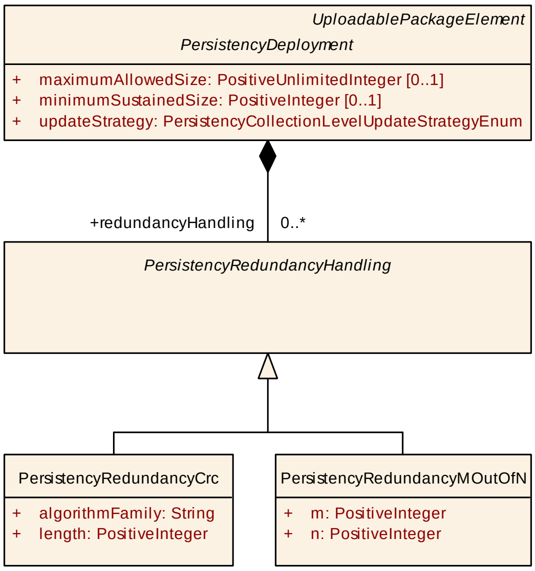


Figure 8.2: Modeling of the abstract base class `PersistencyDeployment`

Class	***PersistencyDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of persistency. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	PersistencyFileArray, PersistencyKeyValueDatabase			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
maximum AllowedSize	PositiveUnlimitedInteger	0..1	attr	The value of this attribute represents the maximum size allowed at deployment time for the enclosing Persistency Deployment.
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size guaranteed at deployment time for the enclosing PersistencyDeployment.
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy. **Tags:** atp.Status=draft

▽

△

Class	PersistencyDeployment (abstract)			
updateStrategy	PersistencyCollectionLevelUpdateStrategyEnum	1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyDeployment as a whole.

Table 8.5: PersistencyDeployment

[TPS_MANI_01206]{DRAFT} **Modeling of redundancy in the context of PersistencyDeployment** ⌈ In contrast to the specification of redundancy on design level where the modeling is more or less reduced to an "on/off" semantics, the deployment level provides the ability to provide a more detailed definition of redundant behavior for both key-value storage as well as files.

This modeling is attached to the abstract base class PersistencyDeployment in order to let both aspects of persistence on the *AUTOSAR adaptive platform* benefit from the existence of meta-class PersistencyRedundancyHandling. ⌋ *(RS_MANI_00027)*

[TPS_MANI_01207]{DRAFT} **Standardized values of attribute PersistencyRedundancyCrc.algorithmFamily** ⌈ The following values of attribute PersistencyRedundancyCrc.algorithmFamily are standardized by AUTOSAR:

- **CRC_J1850**
- **CRC_CCITT_FALSE**
- **CRC_ETHERNET**
- **CRC_0x42F0E1EBA9EA3693**
- **CRC_8H2F**
- **CRC_16ARC**
- **CRC_32P4**

⌋*(RS_MANI_00027)*

[constr_1668]{DRAFT} **Allowed combinations of PersistencyRedundancyCrc.length and algorithmFamily** ⌈ The allowed combinations of PersistencyRedundancyCrc.length and algorithmFamily are documented in Table 8.6. ⌋ *()*

	8	16	32	64
CRC_J1850	x			
CRC_CCITT_FALSE		x		
CRC_ETHERNET			x	

▽

△

CRC_0x42F0E1EBA9EA3693				X
CRC_8H2F	X			
CRC_16ARC		X		
CRC_32P4			X	

Table 8.6: Allowed combinations of PersistencyRedundancyCrc.length and algorithmFamily

Class	*PersistencyRedundancyHandling* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract base class represents a formal description of redundancy. **Tags:** atp.Status=draft			
Base	*ARObject*			
Subclasses	PersistencyRedundancyCrc, PersistencyRedundancyMOutOfN			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 8.7: PersistencyRedundancyHandling

Class	PersistencyRedundancyCrc			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class formally describes the usage of a CRC for the implementation of redundancy. **Tags:** atp.Status=draft			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
algorithmFamily	String	1	attr	This attribute identifies the algorithm family that is used to execute the CRC.
length	PositiveInteger	1	attr	This attribute describes the length of the CRC in the unit bits.

Table 8.8: PersistencyRedundancyCrc

Class	PersistencyRedundancyMOutOfN			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class provides the ability to describe redundancy via an "M out of N" approach. In this case N is the number of copies created and M is the minimum number of identical copies to justify a reliable read access to the data. **Tags:** atp.Status=draft			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
m	PositiveInteger	1	attr	This attribute represents the "M" coordinate in the "M out of N" scheme.
n	PositiveInteger	1	attr	This attribute represents the "N" coordinate in the "M out of N" scheme.

Table 8.9: PersistencyRedundancyMOutOfN

8.2.2 Deployment of Persistent Data

[TPS_MANI_01079]{DRAFT} **Semantics of PersistencyKeyValueDatabase** ⌈
Meta-class PersistencyKeyValueDatabase represents an actual database or similar entity used for persistently storing data. ⌋*(RS_MANI_00027)*

Class	**PersistencyKeyValueDatabase**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to model a key/value data base on deployment level. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistencyKeyValueDatabases			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyDeployment*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
keyValuePair	PersistencyKeyValue Pair	*	aggr	This aggregation represents the key-value-pairs owned by the enclosing PersistencyKeyValueDatabase **Tags:** atp.Status=draft
uri	UriString	0..1	attr	This attribute holds the storage location for the PersistencyKeyValueDatabase / PersistencyFile, e.g. file on the file system.

Table 8.10: PersistencyKeyValueDatabase

[TPS_MANI_01147]{DRAFT} **Semantics of PersistencyKeyValueDatabase.updateStrategy** ⌈ The attribute PersistencyKeyValueDatabase.updateStrategy shall be used to specify the strategy for updating the actual persistent elements.

This update strategy shall be applied to the PersistencyKeyValueDatabase as a whole except for the explicitly modeled keyValuePairs that define their own updateStrategy. ⌋*(RS_MANI_00027)*

[TPS_MANI_01196]{DRAFT} **Semantics of PersistencyKeyValueDatabase.minimumSustainedSize** ⌈ Attribute PersistencyKeyValueDatabase.minimumSustainedSize can be used for the definition of a **minimum amount of storage** that the PersistencyKeyValueDatabase will need to allocate from an integrator's point of view.

It is the responsibility of the underlying platform to make sure that this minimum amount of storage is available at any time. ⌋*(RS_MANI_00027)*

[TPS_MANI_01197]{DRAFT} **Semantics of PersistencyKeyValueDatabase.maximumAllowedSize** ⌈ Attribute PersistencyKeyValueDatabase.maximumAllowedSize can be used for the definition of the **maximum amount of storage** that the PersistencyKeyValueDatabase may need to allocate from an integrator's point of view.

It is the responsibility of the underlying platform to make sure that this maximum amount of storage is available at any time. ⌋*(RS_MANI_00027)*

[TPS_MANI_01144]{DRAFT} **Semantics of PersistencyKeyValuePair** ⌈ Meta-class PersistencyKeyValuePair represents an **entry** to an actual database (formalized by PersistencyKeyValueDatabase) or similar entity used for persistently storing data. ⌋*(RS_MANI_00027)*

Class	PersistencyKeyValuePair			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to formally model a key-value pair in the context of the deployment of persistency. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
initValue	ValueSpecification	1	aggr	This aggregation represents the ability to define an initial value for the value side of the key-value pair. **Tags:** atp.Status=draft
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyKeyValuePair.
valueDataType	AbstractImplementationDataType	1	ref	This reference represents the data type applicable for the value of the key-value pair. **Tags:** atp.Status=draft

Table 8.11: PersistencyKeyValuePair

The modeling of PersistencyKeyValuePair aggregated in the role PersistencyKeyValueDatabase.keyValuePair is optional. It would be possible to use persistency functionality regardless of the existence of keyValuePair.

However, the presence of keyValuePair gives more freedom and ways for the customization of behavior.

[TPS_MANI_01078]{DRAFT} **Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping** ⌈ Meta-class PersistencyPortPrototypeToKeyValueDatabaseMapping has the ability to map a specific PortPrototype referenced in the role portPrototype to a PersistencyKeyValueDatabase referenced in the role keyValueStorage.

The mapping also comprises a reference to meta-class process in order to accommodate for the fact that identical combinations of keyValueStorage and portPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime. ⌋*(RS_MANI_00027)*

[constr_1555]{DRAFT} **Restriction applicable for PersistencyPortPrototypeToKeyValueDatabaseMapping.portPrototype** ⌈ The reference PersistencyPortPrototypeToKeyValueDatabaseMapping.portPrototype shall only be used for a PortPrototype typed by a PersistencyKeyValueDatabaseInterface. ⌋*()*

[TPS_MANI_01155]{DRAFT} **PersistencyKeyValueDatabase.updateStrategy overrides PersistencyKeyValueDatabaseInterface.updateStrategy** ⌈ The value of attribute PersistencyKeyValueDatabase.updateStrategy shall overrule the value of PersistencyKeyValueDatabaseInterface.updateStrategy for any combination of PersistencyKeyValueDatabaseInterface mapped to a PersistencyKeyValueDatabase by means of a PersistencyPortPrototypeToKeyValueDatabaseMapping. ⌋(RS_MANI_00027)

This means that the integrator of the software gets the authority to either agree to the designer's point of view or else overrule the designer's decision based on superior knowledge regarding the integration strategy.

[TPS_MANI_01157]{DRAFT} **Semantics of updateStrategy on collection level** ⌈ The semantics of attribute updateStrategy on collection level is specified in Table 8.12.

The table applies for both the attribute PersistencyKeyValueDatabase.updateStrategy as well as for attribute PersistencyFileArray.updateStrategy. ⌋ (RS_MANI_00027)

updateStrategy	Use Case: Installation	Use Case: Update
delete	irrelevant	delete all elements not contained in current manifest
keepExisting	irrelevant	keep all elements not contained in current manifest

Table 8.12: Semantics of updateStrategy on collection level

[TPS_MANI_01159]{DRAFT} **Semantics of updateStrategy on element level** ⌈ The semantics of attribute updateStrategy on element level is specified in Table 8.13.

The table applies for both attribute PersistencyKeyValuePair.updateStrategy as well as for attribute PersistencyFile.updateStrategy. ⌋(RS_MANI_00027)

updateStrategy	Use Case: Installation	Use Case: Update
delete	don't create	remove
keepExisting	create	do nothing
overwrite	create	replace

Table 8.13: Semantics of updateStrategy on element level

Class	**PersistencyPortPrototypeToKeyValueDatabaseMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a PortPrototype and a key value database used in a persistent storage. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistentPortPrototypeToKeyValueDatabaseMappings			
Base	*ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable, UploadablePackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	**PersistencyPortPrototypeToKeyValueDatabaseMapping**			
keyValue Storage	PersistencyKeyValue Database	1	ref	This reference represents the mapped key-value storage. **Tags:** atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference represents the affected Persistency Port Prototype **Tags:** atp.Status=draft
process	Process	1	ref	This reference represents the process required for context of the mapping. **Tags:** atp.Status=draft

Table 8.14: PersistencyPortPrototypeToKeyValueDatabaseMapping

Please note that typically the existence of PersistencyKeyValueDatabase.keyValuePair depends on the existence of PersistencyKeyValueDatabaseInterface.dataElement.

On the other hand, if a PersistencyKeyValueDatabase contains PersistencyKeyValuePairs that do not correspond to any dataElements of the PersistencyKeyValueDatabaseInterface that is mapped (indirectly) via PersistencyPortPrototypeToKeyValueDatabaseMappingthen those keyValuePairs are created within the PersistencyKeyValueDatabase.

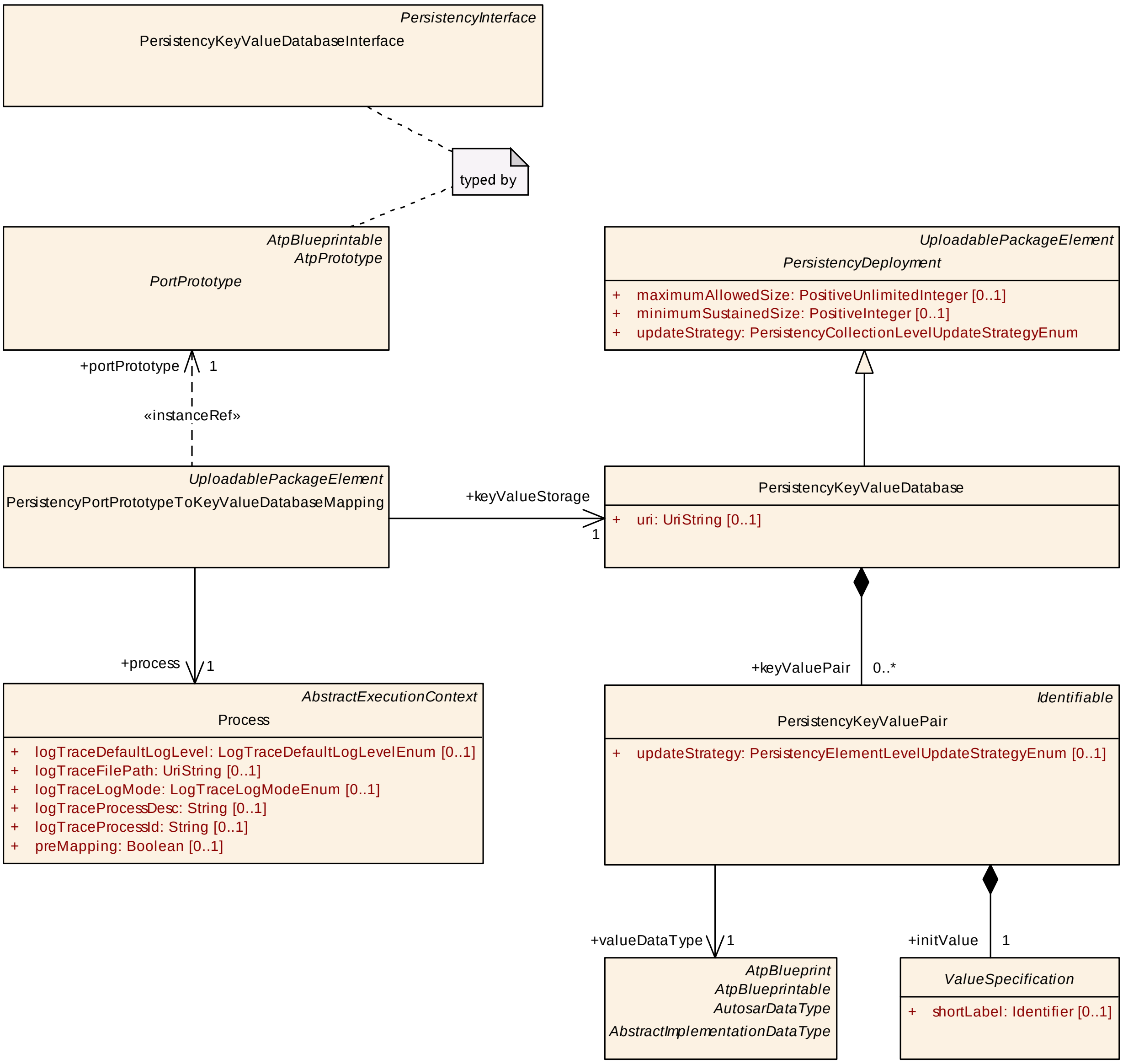


Figure 8.3: Connect a specific PortPrototype to a PersistencyKeyValueDatabase

[TPS_MANI_01146]{DRAFT} **Initial value for PersistencyKeyValuePair** ⌈ It is possible to define an initial value for a given PersistencyKeyValuePair by means of the aggregation of ValueSpecification in the role initValue. ⌋ *(RS_MANI_00027)*

[constr_1554]{DRAFT} **Restriction regarding PersistencyKeyValuePair.initValue** ⌈ The concrete sub-class of ValueSpecification aggregated in the role PersistencyKeyValuePair.initValue shall not (after resolving a possible redirection by means of ConstantReference) be one of the following:

- ApplicationValueSpecification
- ApplicationRuleBasedValueSpecification
- ReferenceValueSpecification

⌋()

[TPS_MANI_01148]{DRAFT} **Semantics of PersistencyKeyValuePair.updateStrategy** ⌈ The attribute PersistencyKeyValuePair.updateStrategy can be used to specify the strategy for updating the actual persistent element that corresponds to PersistencyKeyValuePair. ⌋*(RS_MANI_00027)*

[TPS_MANI_01156]{DRAFT} **PersistencyKeyValuePair.updateStrategy overrides PersistencyKeyValueDatabase.updateStrategy** ⌈ The value specified for PersistencyKeyValuePair.updateStrategy overrides the value of PersistencyKeyValueDatabase.updateStrategy for this specific PersistencyKeyValuePair. ⌋*(RS_MANI_00027)*

[TPS_MANI_01182]{DRAFT} **PersistencyKeyValuePair.updateStrategy overrides PersistencyDataElement.updateStrategy** ⌈ The value of attribute PersistencyKeyValuePair.updateStrategy overrides the value of attribute PersistencyDataElement.updateStrategy ⌋*(RS_MANI_00027)*

This means that the integrator of the software gets the authority to either agree to the designer's point of view or else overrule the designer's decision based on superior knowledge regarding the integration strategy.

[constr_1582]{DRAFT} **PersistencyKeyValuePair.valueDataType shall match to AbstractImplementationDataType for the corresponding PersistencyDataElement** ⌈ Each PersistencyKeyValuePair.valueDataType shall match the AbstractImplementationDataType that either directly or indirectly (via the applicable DataTypeMap) types the corresponding (based on identical values of the respective shortName) PersistencyDataElement. ⌋*()*

[constr_1666]{DRAFT} **References from PersistencyPortPrototypeToKeyValueDatabaseMapping to PersistencyKeyValueDatabase** ⌈ Each PersistencyKeyValueDatabase shall only be referenced by at most one PersistencyPortPrototypeToKeyValueDatabaseMapping. ⌋*()*

8.2.3 Deployment of Files

[TPS_MANI_01150]{DRAFT} **Semantics of PersistencyFileArray** ⌈ A PortPrototype typed by a PersistencyFileProxyInterface actually builds an abstraction for an entire array of files.

This abstraction is also visible in the deployment by means of the existence of the companion meta-class PersistencyFileArray.

This approach allows for the dynamic creation and/or deletion of files during runtime while still keeping the structural model of the file interaction static. ⌋*(RS_MANI_00027)*

Class	PersistencyFileArray			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class comes with the ability to define an array of single files that creates the deployment-side counterpart to a PortPrototype typed by a PersistencyFileProxyInterface. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistencyFileArrays			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyDeployment*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
file	PersistencyFile	*	aggr	This aggregation represents the collection of files aggregated by the PersistencyFileArray. **Tags:** atp.Status=draft
uri	UriString	1	attr	This attribute holds the storage location for the PersistencyFileArray, e.g. a directory on the file system.

Table 8.15: PersistencyFileArray

[TPS_MANI_01151]{DRAFT} **Semantics of PersistencyFileArray.updateStrategy** ⌈ The attribute PersistencyFileArray.updateStrategy shall be used to specify the strategy for updating the actual persistent elements.

This update strategy shall be applied to the PersistencyFileArray as a whole except for the explicitly modeled files that define their own updateStrategy. ⌋ *(RS_MANI_00027)*

At one point, however, it is necessary to boil down the relation of such a PortPrototype typed by a PersistencyFileProxyInterface to individual files and how these individual files are represented on the file system themselves.

This aspect is covered by the modeling of meta-class PersistencyPortPrototypeToFileArrayMapping, as depicted in Figure 8.4.

[TPS_MANI_01080]{DRAFT} **Semantics of PersistencyPortPrototypeToFileArrayMapping** ⌈ Meta-class PersistencyPortPrototypeToFileArrayMapping creates a mapping between a PortPrototype referenced in the role portPrototype to a PersistencyFileArray referenced in the role persistencyFileArray under consideration of a Process referenced in the role process. ⌋ *(RS_MANI_00027)*

Class	PersistencyPortPrototypeToFileArrayMapping
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class represents the ability to define a mapping between an array of files on deployment level to a given PortPrototype. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistentFileProxyToFileMappings
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*

▽

△

Class	**PersistencyPortPrototypeToFileArrayMapping**			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
persistencyFileArray	PersistencyFileArray	1	ref	This reference represents the mapped array of files. **Tags:** atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:** atp.Status=draft
process	Process	1	ref	This reference represents the process required as context for the mapping. **Tags:** atp.Status=draft

Table 8.16: PersistencyPortPrototypeToFileArrayMapping

[TPS_MANI_01154]{DRAFT} **PersistencyFileArray.updateStrategy overrides PersistencyFileProxyInterface.updateStrategy** ⌈ The value of attribute PersistencyFileArray.updateStrategy shall override the value of PersistencyFileProxyInterface.updateStrategy for any combination of PersistencyFileProxyInterface mapped to a PersistencyFileArray by means of a PersistencyPortPrototypeToFileArrayMapping. ⌋*(RS_MANI_00027)*

This means that the integrator of the software gets the authority to either agree to the designer's point of view or else overrule the designer's decision based on superior knowledge regarding the integration strategy.

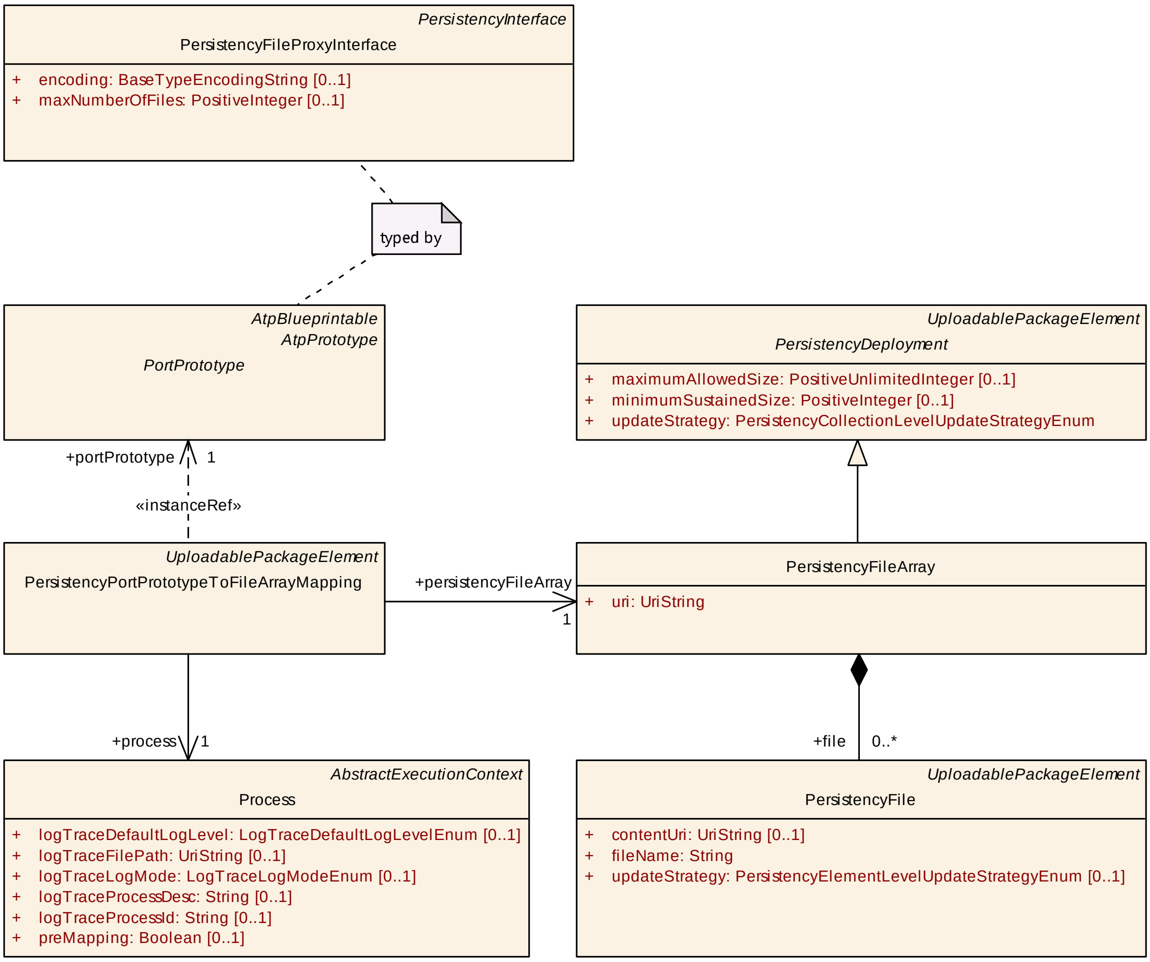


Figure 8.4: Connect a specific PortPrototype to a PersistencyFile

[TPS_MANI_01149]{DRAFT} **Semantics of PersistencyFileArray.file** ⌈ The usage of PersistencyFileArray.file allows for the explicit modeling of elements of the PersistencyFileArray.

The creation of this aggregation is optional. It can be used to define the update strategy and/or initial content of selected files. ⌋*(RS_MANI_00027)*

[constr_1556]{DRAFT} **Restriction applicable for PersistencyPortPrototypeToFileArrayMapping.portPrototype** ⌈ The reference PersistencyPortPrototypeToFileArrayMapping.portPrototype shall only be used for a PortPrototype typed by a PersistencyFileProxyInterface. ⌋*()*

Class	**PersistencyFile**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency

▽

△

Class	PersistencyFile			
Note	This meta-class represents the model of a file as part of the persistency on deployment level. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistencyFiles			
Base	*ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable, UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	1	attr	This attribute holds filename part of the storage location for the PersistencyFile, e.g. file on the file system. **Tags:** atp.Status=draft
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyFile.

Table 8.17: PersistencyFile

[TPS_MANI_01152]{DRAFT} **Semantics of PersistencyFile.updateStrategy** ⌈ The attribute PersistencyFile.updateStrategy can be used to specify the strategy for updating the actual persistent file that corresponds to the model element PersistencyFile. ⌋*(RS_MANI_00027)*

[TPS_MANI_01158]{DRAFT} **PersistencyFile.updateStrategy overrides PersistencyFileArray.updateStrategy** ⌈ The value specified for PersistencyFile.updateStrategy overrides the value of PersistencyFileArray.updateStrategy for this specific PersistencyFile. ⌋*(RS_MANI_00027)*

[TPS_MANI_01183]{DRAFT} **PersistencyFile.updateStrategy overrides PersistencyFileProxy.updateStrategy** ⌈ The value of attribute PersistencyFile.updateStrategy overrides the value of attribute PersistencyFileProxy.updateStrategy. ⌋*(RS_MANI_00027)*

This means that the integrator of the software gets the authority to either agree to the designer's point of view or else overrule the designer's decision based on superior knowledge regarding the integration strategy.

[TPS_MANI_01179]{DRAFT} **Semantics of PersistencyFileProxy.contentUri/PersistencyFile.contentUri vs. PersistencyFileArray.uri and PersistencyFileProxy.fileName/PersistencyFile.fileName** ⌈ Attributes PersistencyFileProxy.contentUri and (after deployment) PersistencyFile.contentUri describe the URI of the file that is used to initialize the PersistencyFile (used during install or update).

On the other hand, the combination of PersistencyFileArray.uri and the PersistencyFileProxy.fileName or (after deployment) PersistencyFile.fileName denote the position of the PersistencyFile in the ECU (used at run-time). ⌋*(RS_MANI_00027)*

[constr_1589]{DRAFT} **Value of `file.fileName`** ⌈ Within the scope of any given `PersistencyFileArray`, the value of all `file.fileName` shall be unique.

A `fileName` is considered unique if there are no other `fileName`s with **exactly** the same sequence of characters[1]. ⌋()

[TPS_MANI_01187]{DRAFT} **Matching pairs of `PersistencyFileProxy` and `PersistencyFile`** ⌈ Matching pairs of `PersistencyFileProxy` and `PersistencyFile` shall be identified by having the identical value of attribute `shortName` within the scope of a `PersistencyFileProxyInterface` (or a `PortPrototype` typed by the `PersistencyFileProxyInterface`) mapped to a `PersistencyFileArray` by means of a `PersistencyPortPrototypeToFileArrayMapping`. ⌋*(RS_MANI_00027)*

[constr_1613]{DRAFT} **File name of matching pairs of `PersistencyFileProxy` and `PersistencyFile`** ⌈ The value of attributes `PersistencyFileProxy.fileName` and `PersistencyFile.fileName` shall be identical for matching pairs (as identified by the application of [TPS_MANI_01187]) of `PersistencyFileProxy` and `PersistencyFile`. ⌋()

[constr_1667]{DRAFT} **References from `PersistencyPortPrototypeToFileArrayMapping` to `PersistencyFileArray`** ⌈ Each `PersistencyFileArray` shall only be referenced by at most one `PersistencyPortPrototypeToFileArrayMapping`. ⌋()

8.3 Platform Health Management Deployment

8.3.1 Overview

This chapter explains the interaction of application software with the Platform Health Management [14].

The `PlatformHealthManagementContribution` allows to describe aspects for the deployment of configuration how the Platform Health Management shall behave during runtime.

[TPS_MANI_03544]{DRAFT} **Definition of `PlatformHealthManagementContribution`** ⌈ The meta-class `PlatformHealthManagementContribution` allows to define a set of configuration entities for the Platform Health Management. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[1] The characters "x" and "X" are not considered as identical characters for this purpose.

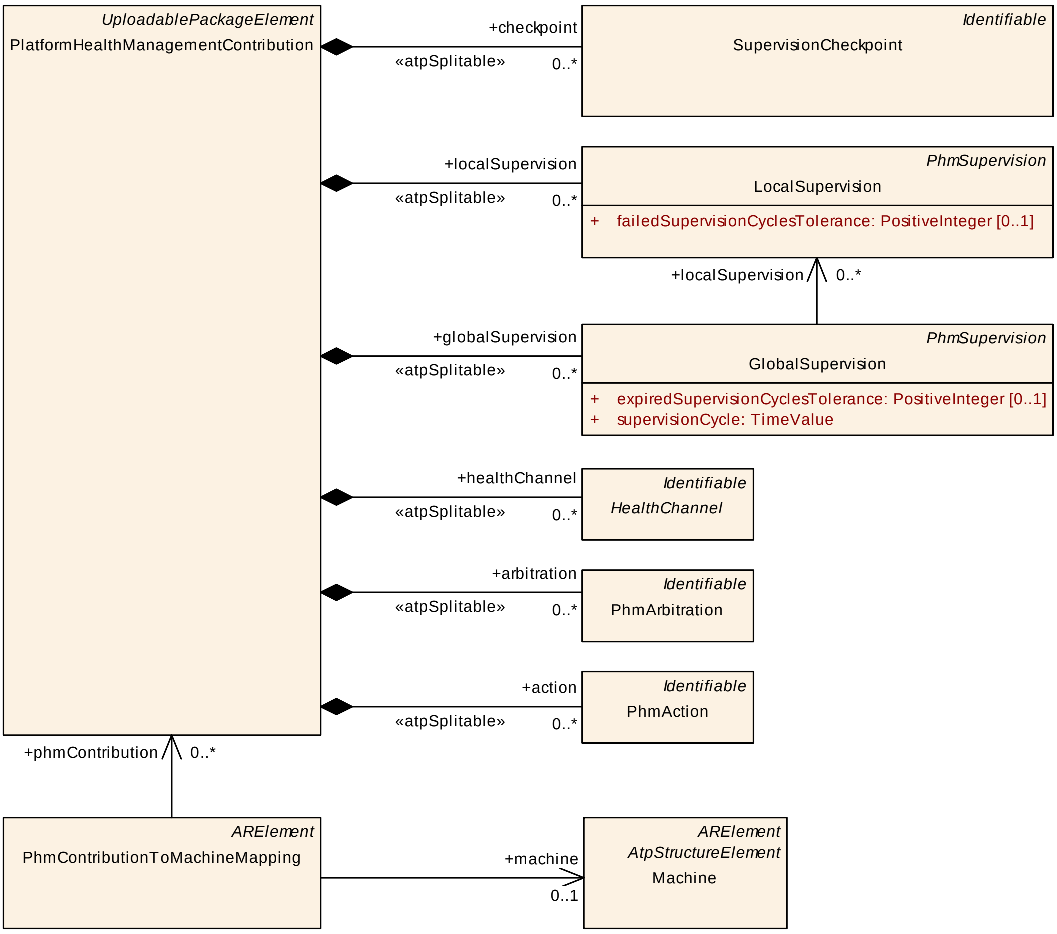


Figure 8.5: Modeling of `PlatformHealthManagementContribution`

The `PlatformHealthManagementContribution` is structured into several aspects which will be described in the following sections:

- Supervision (section 8.3.2)
- Health channels (section 8.3.4)
- Arbitration and Rules (section 8.3.5)
- Actions (section 8.3.6)

Class	**PlatformHealthManagementContribution**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement
Note	This element defines a contribution to the Platform Health Management. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PlatformHealthManagementContributions
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*

▽

△

Class	PlatformHealthManagementContribution			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
action	PhmAction	*	aggr	Collection of Actions and ActionLists in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=60
arbitration	PhmArbitration	*	aggr	Collection of Arbitrations in the context of a Platform HealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=50
checkpoint	SupervisionCheckpoint	*	aggr	Collection of checkpoints in the context of a Platform HealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=10
global Supervision	GlobalSupervision	*	aggr	Collection of GlobalSupervisions in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=30
healthChannel	HealthChannel	*	aggr	Collection of HealthChannels in the context of a Platform HealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=40
local Supervision	LocalSupervision	*	aggr	Collection of LocalSupervisions in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=20

Table 8.18: PlatformHealthManagementContribution

[TPS_MANI_03502]{DRAFT} **Enabling of PlatformHealthManagementContribution on a Machine** ⌈ To enable an instance of PlatformHealthManagementContribution on a specific Machine the PlatformHealthManagementContribution shall be mapped to the Machine via a PhmContributionToMachineMapping. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**PhmContributionToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element associates one or more PlatformHealthManagementContributions with a Machine. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PhmContributionToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the PhmContributionToMachineMapping. **Tags:** atp.Status=draft
phm Contribution	PlatformHealth Management Contribution	*	ref	This reference identifies one or more PlatformHealth ManagementContributions in the context of a Phm ContributionToMachineMapping. **Tags:** atp.Status=draft

Table 8.19: PhmContributionToMachineMapping

An application software can define the usage of several Platform Health Management supervisions (see chapter 3.10.2) and health channels (see chapter 3.10.3). In order to define the interaction between the application software and the Platform Health Management the `PlatformHealthManagementContribution` creates its own representations of the `RPortPrototype`s typed by the `PhmSupervisedEntityInterface` and `PhmHealthChannelInterface` and creates relations to the application software `RPortPrototype`s (see figure 8.6).

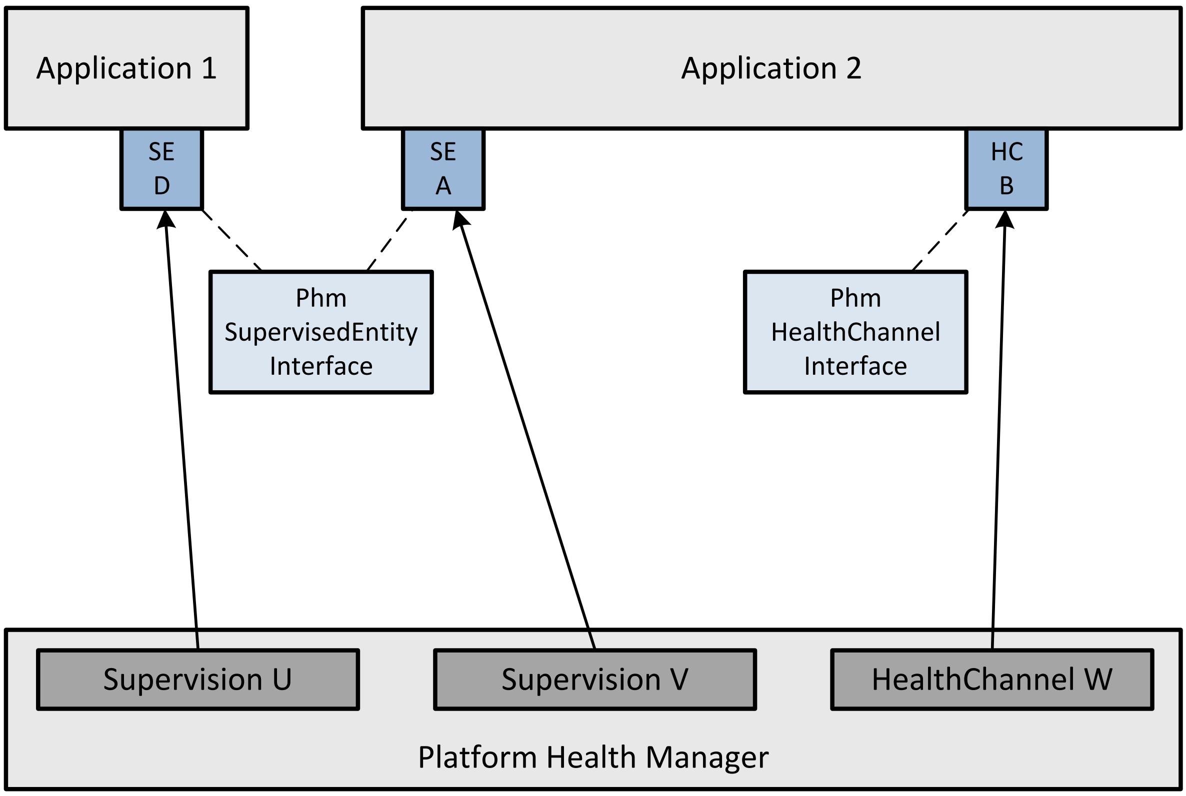


Figure 8.6: Interaction of application software with the platform health manager

In chapter 3.10.2 it is explained that the application software just calls methods in the context of the respective `RPortPrototype`s to interact with the Platform Health Management. From the application developer these methods have no addressing information, because the identity of the `RPortPrototype` is the identification in the scope of the application software.

The deployed structure (according to figure 8.5) however requires more information when an `API` at the Platform Health Manager is called, namely:

- `RPortPrototype.shortName` i.e. `InstanceSpecifier`
- `Process` identification during runtime.

These additional arguments have to be injected to the `API` by the implementation of the interaction between the software component and the Platform Health Management (which implements the relations from figure 8.5). The order of this argument injection is determined by the specification of the Platform Health Management APIs.

8.3.2 Supervision deployment

In the application design chapter of this document the declaration of supervised entities and checkpoints has been described (see section 3.10.2). These declarations provide the view on supervision from the application software code point. Since the application `Executable` can be started multiple times (via individual `Process`es) the configuration of the Platform Health Management needs to cope with these individual `Executable` instances.

[TPS_MANI_03503]{DRAFT} **Applicability of checkpoints to a specific `Process`** ⌈ The reference `SupervisionCheckpoint.process` defines to which specific `Process` this `SupervisionCheckpoint` definition shall be applied to. ⌋ *(RS_MANI_00023, RS_MANI_00032)*

This means that only if a `PhmCheckpoint` is reported from the context of this `Process` it is considered to be this `SupervisionCheckpoint`.

For the Platform Health Management supervision to take effect it is required to define the instance the application is executed in, thus the reference to a `Process` has to be taken into account. In the model the `Process` also defines under which conditions (`StateDependentStartupConfig`) and with which arguments (`StartupOption`) the `Executable` will be started.

For the configuration of the Platform Health Management the definition of `SupervisionCheckpoint` is used to stand in for the corresponding `PhmCheckpoint` including the execution context of the respective `Process`.

[TPS_MANI_03505]{DRAFT} **Existence of SupervisionCheckpoint** ⌈ For each PhmCheckpoint in the scope of a RPortPrototype typed by a PhmSupervisedEntityInterface in the application definition there may be a SupervisionCheckpoint defined. The correspondence of the two is defined by the instance reference SupervisionCheckpoint.phmCheckpoint ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03506]{DRAFT} **Optionality of SupervisionCheckpoint** ⌈ It is not required that every PhmSupervisedEntityInterface or PhmCheckpoint used in the context of the application definition eventually has a corresponding SupervisionCheckpoint defined. There may be cases where the application software reports some checkpoints but they are not considered for a specific supervision. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03515]{DRAFT} **Expiration tolerance for LocalSupervision** ⌈ The attribute LocalSupervision.failedSupervisionCyclesTolerance defines how many supervision cycles an incorrect supervision is maintained in the state *failed* before it is considered *expired*. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**LocalSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a LocalSupervision in the context of platform health management contribution. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, PhmSupervision, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
aliveSupervision	AliveSupervision	*	aggr	Collection of AliveSupervisions in the context of this LocalSupervision. **Tags:** atp.Status=draft
deadlineSupervision	DeadlineSupervision	*	aggr	Collection of DeadlineSupervisions in the context of this LocalSupervision. **Tags:** atp.Status=draft
failedSupervisionCyclesTolerance	PositiveInteger	0..1	attr	Defines the acceptable amount of cycles with FAILED supervision status of this LocalSupervision before it is considered EXPIRED. **Tags:** atp.Status=draft
logicalSupervision	LogicalSupervision	*	aggr	Collection of LogicalSupervisions in the context of this LocalSupervision. **Tags:** atp.Status=draft
transition	CheckpointTransition	*	aggr	Collection of CheckpointTransitions in the context of this LocalSupervision. **Tags:** atp.Status=draft

Table 8.20: LocalSupervision

Class	**SupervisionCheckpoint**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element contains an instance reference to a RPortPrototype representing a checkpoint for Platform Health Management. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
phmCheckpoint	PhmCheckpoint	0..1	iref	Instance reference to the PhmCheckpoint defined in the context of a PortInterface. **Tags:** atp.Status=draft
process	Process	1	ref	Reference to the Process this checkoint shall be monitored. **Tags:** atp.Status=draft

Table 8.21: SupervisionCheckpoint

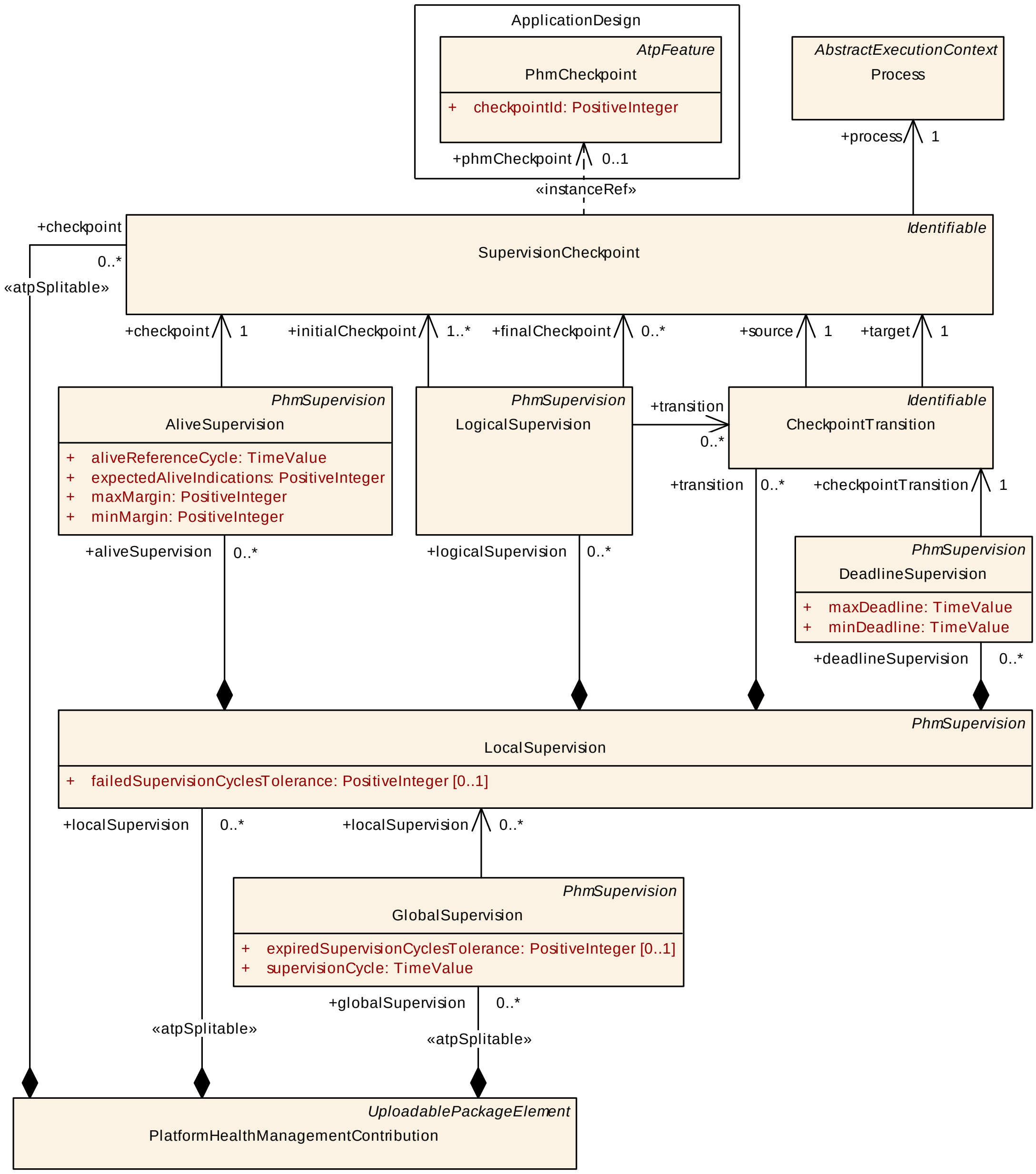


Figure 8.7: Modeling of LocalSupervision

8.3.2.1 AliveSupervision definition

In the scope of a LocalSupervision an AliveSupervision can be defined for a specific SupervisionCheckpoint. LocalSupervision can be used to define in which timing boundaries one specific checkpoint shall be monitored.

[TPS_MANI_03508]{DRAFT} **Definition of an AliveSupervision for a SupervisionCheckpoint** ⌈ An AliveSupervision definition provides attributes to configure the supervision of the referenced SupervisionCheckpoint.

- aliveReferenceCycle defines the time base used to monitor the reporting of this specific SupervisionCheckpoint
- expectedAliveIndications defines the number of indications which shall be observed during the time period defined by aliveReferenceCycle
- minMargin and maxMargin define the acceptable deviation from the expectedAliveIndications within the time period defined by aliveReferenceCycle

⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3539]{DRAFT} **Only one AliveSupervision per SupervisionCheckpoint** ⌈ A SupervisionCheckpoint shall only be referenced up to once by an AliveSupervision in the role checkpoint. ⌋*()*

Class	**AliveSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines an AliveSupervision for one checkpoint. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, PhmSupervision, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
aliveReferenceCycle	TimeValue	1	attr	Time period at which the Alive Supervision mechanism compares the amount of received Alive Indications for the SupervisionCheckpoint against the expectedAliveIndications. **Tags:** atp.Status=draft
checkpoint	SupervisionCheckpoint	1	ref	Reference to a checkpoint in the context of Alive Supervision. **Tags:** atp.Status=draft
expectedAliveIndications	PositiveInteger	1	attr	Defines the amount of expected Alive Indications of the SupervisionCheckpoint within the aliveReferenceCycle. **Tags:** atp.Status=draft
maxMargin	PositiveInteger	1	attr	Defines the amount of Alive Indications of the Supervision Checkpoint that are acceptable to be additional to the expectedAliveIndications within the aliveReferenceCycle. **Tags:** atp.Status=draft
minMargin	PositiveInteger	1	attr	Defines the amount of Alive Indications of the Supervision Checkpoint that are acceptable to be missing to the expectedAliveIndications within the aliveReferenceCycle. **Tags:** atp.Status=draft

Table 8.22: AliveSupervision

8.3.2.2 CheckpointTransition definition

For the definition of further supervision strategies the need to first define possible CheckpointTransitions between SupervisionCheckpoints arises. Since the application software design does not provide any transition definition between checkpoints it is essential to define possible CheckpointTransitions.

The definition of CheckpointTransitions is done in the scope of the LocalSupervision and can be used by the LogicalSupervision and DeadlineSupervision.

[TPS_MANI_03509]{DRAFT} **Definition of a CheckpointTransition** ⌈ A CheckpointTransition defines one possible transition from the source SupervisionCheckpoint to the target SupervisionCheckpoint. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**CheckpointTransition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines one transition between two checkpoints. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
source	SupervisionCheckpoint	1	ref	Reference to the source checkpoint for this transition. **Tags:** atp.Status=draft
target	SupervisionCheckpoint	1	ref	Reference to the target checkpoint for this transition. **Tags:** atp.Status=draft

Table 8.23: CheckpointTransition

8.3.2.3 LogicalSupervision definition

The LogicalSupervision defines a supervision graph of allowed CheckpointTransitions which is monitored by the Platform Health Management without any timing considerations, just the order of reported checkpoints is considered for the monitoring.

[constr_3540]{DRAFT} **SupervisionCheckpoint in supervision graph** ⌈ Each SupervisionCheckpoint shall only be part of one supervision graph. ⌋*()*

When a SupervisionCheckpoint belonging to the supervision graph is reported to the Platform Health Management where there is no CheckpointTransition defined from the last reported SupervisionCheckpoint as source to the current reported SupervisionCheckpoint as target, this situation violates the LogicalSupervision.

[TPS_MANI_03510]{DRAFT} **Definition of LogicalSupervision** ⌈ A LogicalSupervision defines relations between SupervisionCheckpoints which form a directed graph from one or more initialCheckpoint SupervisionCheckpoints through a set of CheckpointTransitions defined by collection of transitions to one or more finalCheckpoint SupervisionCheckpoints. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	LogicalSupervision			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines a LogicalSupervision graph consisting of transitions, initial- and final checkpoints. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, PhmSupervision, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
finalCheckpoint	SupervisionCheckpoint	*	ref	Reference to the final Checkpoint(s) for this Logical Supervision. **Tags:** atp.Status=draft xml.sequenceOffset=20
initialCheckpoint	SupervisionCheckpoint	1..*	ref	Reference to the initial Checkpoint(s) for this Logical Supervision. **Tags:** atp.Status=draft xml.sequenceOffset=10
transition	CheckpointTransition	*	ref	Reference to the transitions for this LogicalSupervision. **Tags:** atp.Status=draft xml.sequenceOffset=30

Table 8.24: LogicalSupervision

8.3.2.4 `DeadlineSupervision` definition

The `DeadlineSupervision` defines timing attributes for one specific `CheckpointTransition`.

[TPS_MANI_03511]{DRAFT} **Definition of `DeadlineSupervision`** ⌈ A `DeadlineSupervision` defines timing attributes which are monitored by the Platform Health Management for one specific `CheckpointTransition`. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	DeadlineSupervision			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines an DeadlineSupervision for one transition. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, PhmSupervision, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
checkpoint Transition	CheckpointTransition	1	ref	Reference to the transition in the context of a Deadline Supervision. **Tags:** atp.Status=draft
maxDeadline	TimeValue	1	attr	Defines the longest time span before which the deadline is considered to be met for transition. **Tags:** atp.Status=draft

▽

△

Class	DeadlineSupervision			
minDeadline	TimeValue	1	attr	Defines the shortest time span after which the deadline is considered to be met for transition. **Tags:** atp.Status=draft

Table 8.25: DeadlineSupervision

8.3.3 Global supervision entity deployment

The GlobalSupervision definition of supervision for the Platform Health Management is a second level supervision which takes the status of one or several LocalSupervisions (with their respective AliveSupervisions, LogicalSupervisions, and DeadlineSupervisions) and aggregates the individual states of these LocalSupervisions into one global supervision status (see also figure 8.7).

[TPS_MANI_03513]{DRAFT} **Collection of LocalSupervisions into a global supervision** ⌈ All referenced LocalSupervisions in the scope of GlobalSupervision.localSupervision shall be taken into the aggregation of the status of the GlobalSupervision. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03512]{DRAFT} **Applicability of global supervision without Process context** ⌈ The referenced LocalSupervisions contributing to a specific GlobalSupervision may refer to SupervisionCheckpoints where each SupervisionCheckpoint may refer to a different Process. ⌋*(RS_MANI_00023, RS_MANI_00032)*

With [TPS_MANI_03512] the GlobalSupervision and the reference GlobalSupervision.localSupervision can be used to establish two use-cases:

- compose the status of one Executable instance (Process) in case all referenced LocalSupervisions are defined with the same Process context
- compose the status of several Executable instances (Processes) in case the referenced LocalSupervisions are defined with (partially) different Process contexts.

[constr_3537]{DRAFT} **LocalSupervision referenced once in the context of a GlobalSupervision** ⌈ Any LocalSupervision shall be referenced at most once by a GlobalSupervision in the role GlobalSupervision.localSupervision. ⌋*()*

[constr_3537] associates a LocalSupervision to up to one GlobalSupervision.

[TPS_MANI_03514]{DRAFT} **Expiration tolerance for GlobalSupervision** ⌈ The attribute GlobalSupervision.expiredSupervisionCyclesTolerance defines how many supervision cycles this incorrect global supervision is maintained in the state *expired* before it is considered *stopped*. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03552]{DRAFT} **Supervision cycle for GlobalSupervision** ⌈ The attribute GlobalSupervision.supervisionCycle defines at which rate the GlobalSupervision shall be monitored. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**GlobalSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a collection of LocalSupervisions in order to provide a aggregated supervision state. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
expiredSupervisionCyclesTolerance	PositiveInteger	0..1	attr	Defines the acceptable amount of cycles with EXPIRED supervision status of this GlobalSupervision before it is considered STOPPED. **Tags:** atp.Status=draft
localSupervision	LocalSupervision	*	ref	Reference to the LocalSupervisions which are used to derive the status of this GlobalSupervision. **Tags:** atp.Status=draft
supervisionCycle	TimeValue	1	attr	Defines at which cycle the GlobalSupervision shall be executed.

Table 8.26: GlobalSupervision

8.3.4 Health channel deployment

The HealthChannel is used as an abstraction to the Platform Health Management input for the arbitration and rule evaluation (see chapter 8.3.5).

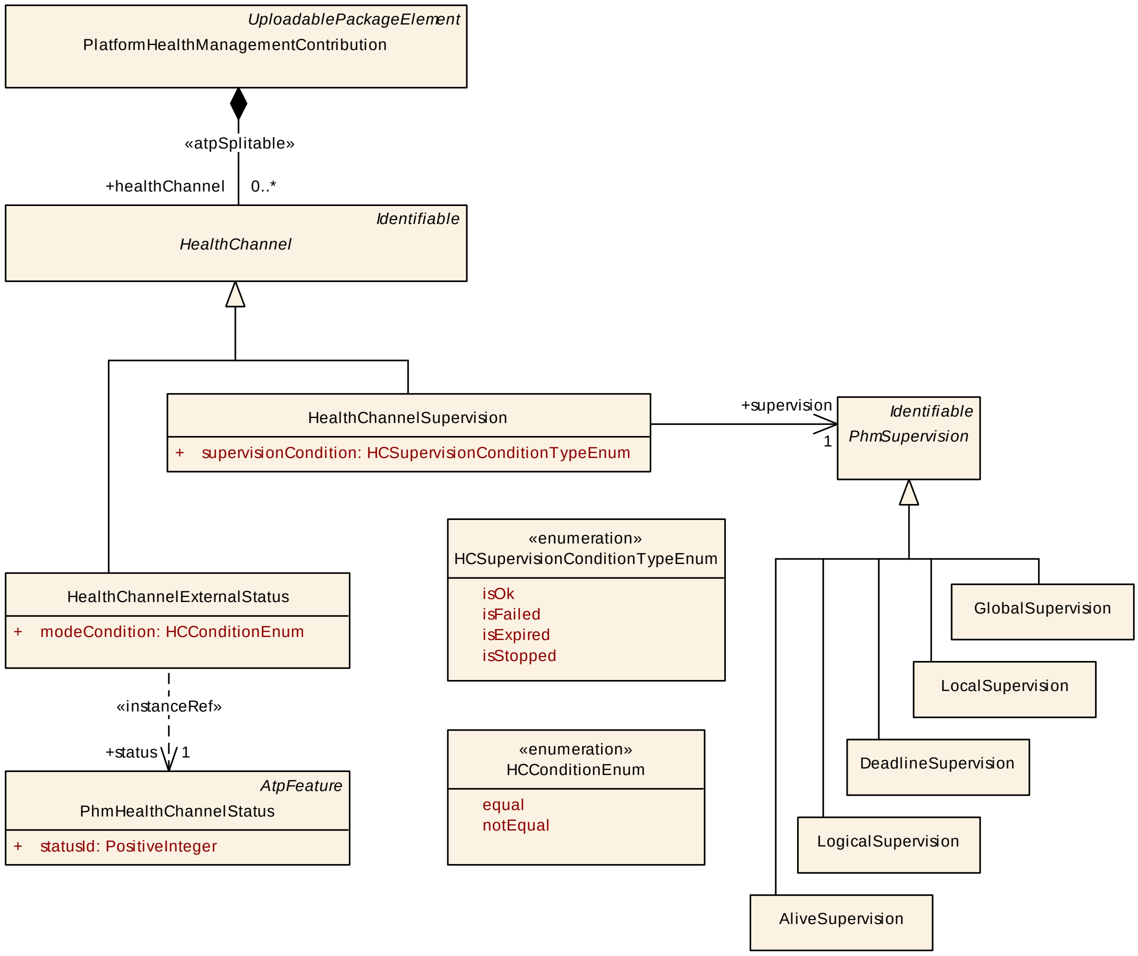


Figure 8.8: Modeling of HealthChannel

Class	***HealthChannel*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines the source of a health channel. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	HealthChannelExternalStatus, HealthChannelSupervision			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.27: HealthChannel

The specialized use-cases for HealthChannels are describe in the following sections.

8.3.4.1 Supervision health channel deployment

The HealthChannelSupervision is used to compare the status of a checkpoint supervision with a constant status definition and provide the result as input to the Platform Health Management arbitration engine.

The input for the checkpoint supervision can be one of the sub classes of PhmSupervision, namely

- GlobalSupervision
- LocalSupervision
- DeadlineSupervision
- LogicalSupervision
- AliveSupervision

[TPS_MANI_03516]{DRAFT} **Condition evaluation for HealthChannelSupervision** ⌈ The status of the PhmSupervision which is referenced in the role supervision will be compared to the constant status provided in supervisionCondition. The result of this comparison is then the result of the HealthChannelSupervision evaluation. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**HealthChannelSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a health channel representing the status of a GlobalSupervision. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, HealthChannel, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
supervision	PhmSupervision	1	ref	Reference to the supervision as source for the health channel. **Tags:** atp.Status=draft
supervision Condition	HCSupervision ConditionTypeEnum	1	attr	Defines which condition shall trigger this health channel wrt. the referenced GlobalSupervision. **Tags:** atp.Status=draft

Table 8.28: HealthChannelSupervision

Enumeration	**HCSupervisionConditionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management
Note	Defines the possible conditions which can be evaluated in the scope of a GlobalSupervision. **Tags:** atp.Status=draft
Literal	***Description***
isExpired	**Tags:** atp.EnumerationValue=2

▽

△

Enumeration	**HCSupervisionConditionTypeEnum**
isFailed	**Tags:** atp.EnumerationValue=1
isOk	**Tags:** atp.EnumerationValue=0
isStopped	**Tags:** atp.EnumerationValue=3

Table 8.29: HCSupervisionConditionTypeEnum

8.3.4.2 External health channel deployment

The HealthChannelExternalStatus is used to compare a reported status to a constant status declaration and provide the result as input to the Platform Health Management arbitration engine.

[TPS_MANI_03545]{DRAFT} **Existence of HealthChannelExternalStatus** ⌈ For each RPortPrototype typed by a PhmHealthChannelInterface there may be a HealthChannelExternalStatus defined. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03546]{DRAFT} **Definition of reported health status RPortPrototype** ⌈ The RPortPrototype typed by a PhmHealthChannelInterface is used to report the status of a health channel by the application software. This specific RPortPrototype is defined as the contextRPortPrototype of the instance reference HealthChannelExternalStatus.status. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03517]{DRAFT} **Condition evaluation for HealthChannelExternalStatus** ⌈ The reported value of the HealthChannelExternalStatus according to [TPS_MANI_03546] will be compared to the constant status provided in status. The modeCondition defines whether it shall be compared for equality or non-equality. The result of this comparison is then the result of the HealthChannelExternalStatus evaluation. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03553]{DRAFT} **Applicability of health channel to a specific Process** ⌈ The reference HealthChannelExternalStatus.process defines to which specific Process this HealthChannelExternalStatus definition shall be applied to. ⌋*(RS_MANI_00023, RS_MANI_00032)*

This means that only if a PhmHealthChannelStatus is reported from the context of this Process it is considered to be this HealthChannelExternalStatus.

Class	**HealthChannelExternalStatus**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement
Note	This element defines a health channel representing the status of an external health channel. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft

▽

△

Class	**HealthChannelExternalStatus**			
Base	*ARObject, HealthChannel, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
modeCondition	HCConditionEnum	1	attr	Defines which condition shall trigger this health channel wrt. the referenced mode. **Tags:** atp.Status=draft
process	Process	1	ref	Defines the Process this Health Channel shall be monitored. **Tags:** atp.Status=draft
status	PhmHealthChannel Status	1	iref	Defines the status to be compared with for the Health Channel. **Tags:** atp.Status=draft

Table 8.30: HealthChannelExternalStatus

8.3.5 Arbitration and rule deployment

The `PhmArbitration` defines the expressions and rules to calculate a logical statement from a set of input `HealthChannel`s. The results of these calculations are used to define the triggering of specific actions by the Platform Health Management (see chapter 8.3.6).

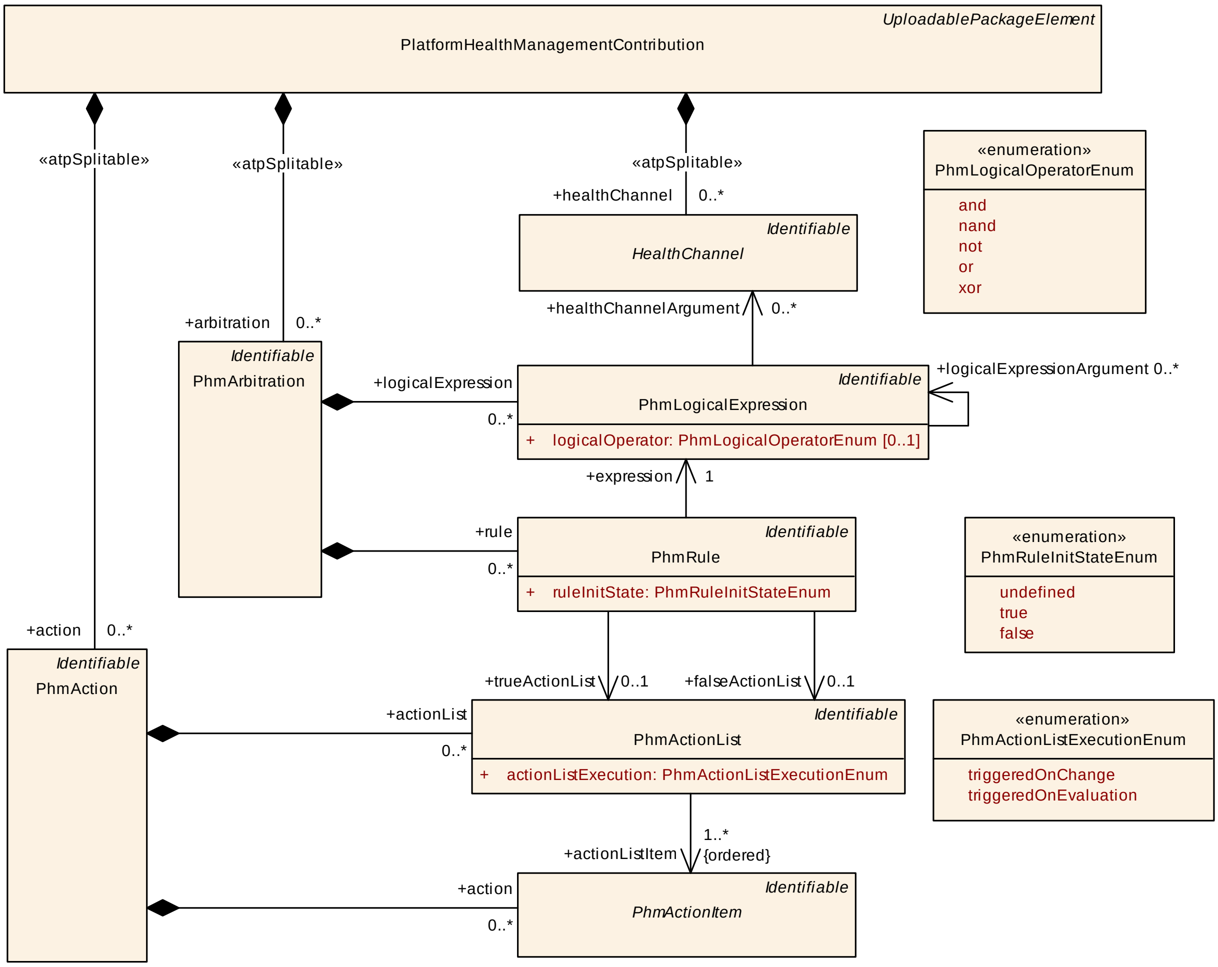


Figure 8.9: Modeling of PhmArbitration

[TPS_MANI_03518]{DRAFT} **PhmLogicalExpression definition** ⌈ A PhmLogicalExpression defines one logicalOperator which will be applied to a set of inputs defined by the healthChannelArgument and logicalExpressionArgument. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Thus the result of a PhmLogicalExpression can again be used as the input to another PhmLogicalExpression.

There are some concerns which need to be formalized at a later point in time to make the definition of PhmLogicalExpressions unambiguous:

- using more than 2 inputs for a PhmLogicalExpression may lead to ambiguous definitions for some logicalOperators
- the inputs to the PhmLogicalExpression are not ordered
- cyclic or recursive definition of PhmLogicalExpressions have to be excluded
- ...

Class	**PhmArbitration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines logical expressions and rules to be evaluated by the platform health management. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
logical Expression	PhmLogicalExpression	*	aggr	Collection of LogicalExpressions in the context of an Arbitration. **Tags:** atp.Status=draft
rule	PhmRule	*	aggr	Collection of rules in the context of an Arbitration. **Tags:** atp.Status=draft

Table 8.31: PhmArbitration

Class	**PhmLogicalExpression**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a logical expression with an arbitrary number of arguments. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
healthChannel Argument	HealthChannel	*	ref	Reference to the HealthChannels which shall be considered for the evaluation of the LogicalExpression. **Tags:** atp.Status=draft
logical Expression Argument	PhmLogicalExpression	*	ref	Reference to another LogicalExpression which shall be considered in the evaluation of this LogicalExpression. **Tags:** atp.Status=draft
logicalOperator	PhmLogicalOperator Enum	0..1	attr	Definition of the operator to be applied to this Logical Expression. **Tags:** atp.Status=draft

Table 8.32: PhmLogicalExpression

Enumeration	**PhmLogicalOperatorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management
Note	Definition of logical expression operators. **Tags:** atp.Status=draft
Literal	*Description*
and	**Tags:** atp.EnumerationValue=0
nand	**Tags:** atp.EnumerationValue=1
not	**Tags:** atp.EnumerationValue=2
or	**Tags:** atp.EnumerationValue=3
xor	**Tags:** atp.EnumerationValue=4

Table 8.33: PhmLogicalOperatorEnum

The result of a `PhmLogicalExpression` is taken as input to a `PhmRule` where it is decided whether and which reaction has to be performed.

[TPS_MANI_03519]{DRAFT} **PhmRule definition** ⌈ A PhmRule takes the result of exactly one PhmLogicalExpression and defines the handling of a reaction based on the result of the PhmLogicalExpression:

- if the PhmLogicalExpression evaluates to *true* the PhmActionList referenced in the role trueActionList will be indicated for execution
- if the PhmLogicalExpression evaluates to *false* the PhmActionList referenced in the role falseActionList will be indicated for execution

Whether an PhmActionList is actually executed is depending on the setting of actionListExecution (see [TPS_MANI_03520]). ⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3527]{DRAFT} **PhmLogicalExpression referenced by one PhmRule** ⌈ Each PhmLogicalExpression shall only be referenced by up to one PhmRule in the role PhmRule.expression. ⌋*()*

Class	**PhmRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a rule for the platform health management. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
expression	PhmLogicalExpression	1	ref	Reference to the logical expression that is evaluated for this rule. **Tags:** atp.Status=draft
falseActionList	PhmActionList	0..1	ref	Reference to the action list which shall be executed when the rule evaluates to FALSE. **Tags:** atp.Status=draft
ruleInitState	PhmRuleInitStateEnum	1	attr	Defines the initial state of this rule. **Tags:** atp.Status=draft
trueActionList	PhmActionList	0..1	ref	Reference to the action list which shall be executed when the rule evaluates to TRUE. **Tags:** atp.Status=draft

Table 8.34: PhmRule

Enumeration	**PhmRuleInitStateEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management
Note	Definition of initial states for rules. **Tags:** atp.Status=draft
Literal	***Description***
false	**Tags:** atp.EnumerationValue=2
true	**Tags:** atp.EnumerationValue=1

▽

△

Enumeration	**PhmRuleInitStateEnum**
undefined	**Tags:** atp.EnumerationValue=0

Table 8.35: PhmRuleInitStateEnum

The PhmActionList collects an ordered list of PhmActionItems to be executed when the PhmActionList is executed. Whether an PhmActionList is actually executed is defined by the actionListExecution.

[TPS_MANI_03520]{DRAFT} **Execution of PhmActionList with actionListExecution=triggeredOnEvaluation** ⌈When a PhmRule indicates the execution of an PhmActionList with actionListExecution=triggeredOnEvaluation this PhmActionList is unconditionally executed every time the PhmRule is evaluated.⌋ *(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03521]{DRAFT} **Execution of PhmActionList with actionListExecution=triggeredOnChange** ⌈When a PhmRule indicates the execution of an PhmActionList with actionListExecution=triggeredOnChange this PhmActionList is only executed when the previous state of the PhmRule was different from the current state.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**PhmActionList**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines an action list for the platform health management. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
actionList Execution	PhmActionListExecution Enum	1	attr	Defines the execution semantics for this action list. **Tags:** atp.Status=draft
actionList Item (ordered)	PhmActionItem	1..*	ref	Ordered reference to the action items to be executed in the scope of this action list. **Tags:** atp.Status=draft

Table 8.36: PhmActionList

Enumeration	**PhmActionListExecutionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management
Note	Definition of execution semantics for action lists. **Tags:** atp.Status=draft
Literal	*Description*
triggeredOnChange	Actions shall only be executed when the evaluation result of the corresponding rule changes. **Tags:** atp.EnumerationValue=0

▽

△

Enumeration	**PhmActionListExecutionEnum**
triggeredOn Evaluation	Actions shall be executed every time the evaluation of the corresponding rule is done. **Tags:** atp.EnumerationValue=1

Table 8.37: PhmActionListExecutionEnum

Class	***PhmActionItem*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines one possible action for the platform health management. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ApplicationModeRequestPhmActionItem, FunctionGroupModeRequestPhmActionItem, PlatformPhmActionItem, ProcessPhmActionItem, WatchdogPhmActionItem			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.38: PhmActionItem

8.3.6 Action deployment

Actions are executed in the scope of an `PhmActionList` in a well defined order. The specific subtypes of actions are described below.

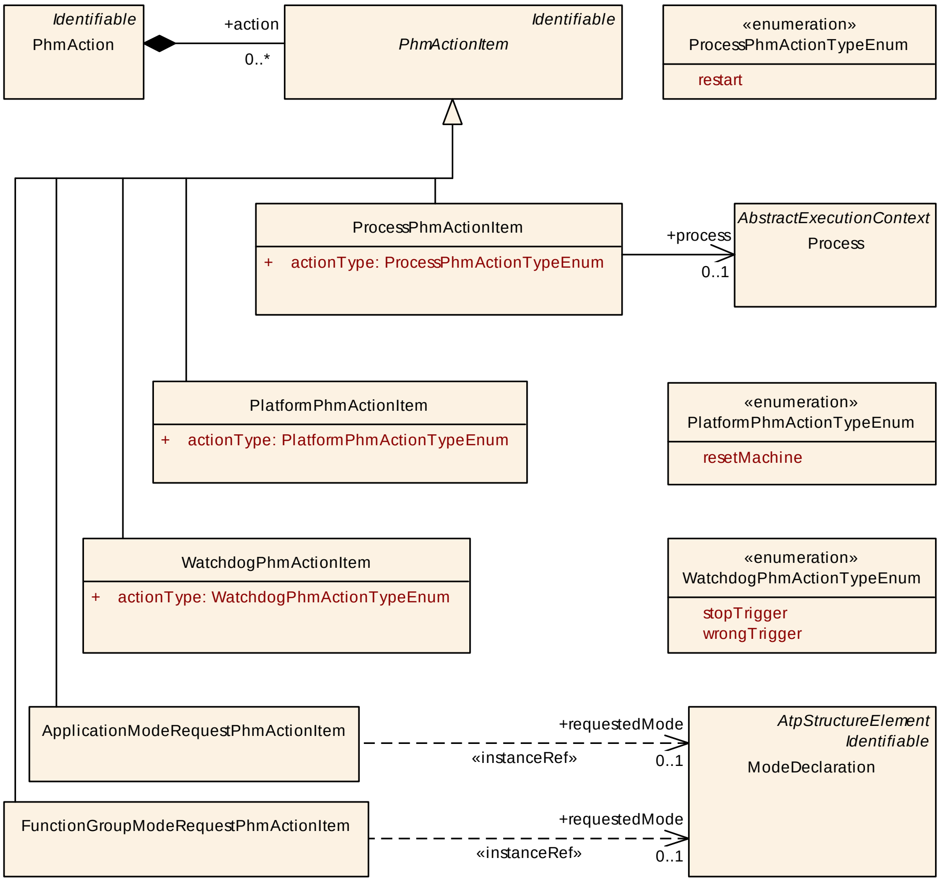


Figure 8.10: Modeling of actions

8.3.6.1 Process action deployment

The ProcessPhmActionItem defines an action which is specific to an instance of an application software (represented by a Process). The action will be forwarded to the Execution Management [17] by the Platform Health Management.

[TPS_MANI_03522]{DRAFT} **Definition of actions for application software** ⌈ The ProcessPhmActionItem defines an action for a specific Process. The action can be to restart the Process. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**ProcessPhmActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines the action to be performed for one specific Process. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, PhmActionItem, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	**ProcessPhmActionItem**			
actionType	ProcessPhmActionType Enum	1	attr	Defines the action be performed on the referenced Process. **Tags:** atp.Status=draft
process	Process	0..1	ref	Reference to the process which represents the application instance. **Tags:** atp.Status=draft

Table 8.39: ProcessPhmActionItem

Enumeration	**ProcessPhmActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management
Note	Definition of available actions to be applied to a Process. **Tags:** atp.Status=draft
Literal	*Description*
restart	Perform a restart of the referenced Process. **Tags:** atp.EnumerationValue=0

Table 8.40: ProcessPhmActionTypeEnum

8.3.6.2 Platform action deployment

The `PlatformPhmActionItem` defines an action which is targeting the whole Platform Instance. The action will be forwarded to the Execution Management [17] by the Platform Health Management.

[TPS_MANI_03523]{DRAFT} **Definition of actions for Platform Instance** ⌈ The `PlatformPhmActionItem` defines an action for the Platform Instance. Different kinds of possible reset strategies are defined in the attribute `PlatformPhmActionItem.actionType`. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**PlatformPhmActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines the action to be performed for this platform instance. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, PhmActionItem, Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
actionType	PlatformPhmActionType Enum	1	attr	Defines the action be performed on this platform instance. **Tags:** atp.Status=draft

Table 8.41: PlatformPhmActionItem

Enumeration	**PlatformPhmActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management
Note	Definition of available actions to be applied to a platform instance. **Tags:** atp.Status=draft
Literal	*Description*
resetMachine	Initiate a reset of the platform instance. **Tags:** atp.EnumerationValue=0

Table 8.42: PlatformPhmActionTypeEnum

8.3.6.3 Watchdog action deployment

The WatchdogPhmActionItem defines an action which is specific to a Watchdog.

[TPS_MANI_03524]{DRAFT} **Definition of actions for Watchdog** ⌈ The WatchdogPhmActionItem defines an action for the Watchdog. One WatchdogPhmActionItem.actionType for the watchdog is to stop triggering of the watchdog (stopTrigger). Another possible actionType for the watchdog is to deliberately cause a wrong triggering of the watchdog (wrongTrigger). ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**WatchdogPhmActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines the action be performed on the watchdog. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmActionItem*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
actionType	WatchdogPhmActionTypeEnum	1	attr	Defines the action to be performed on the watchdog. **Tags:** atp.Status=draft

Table 8.43: WatchdogPhmActionItem

Enumeration	**WatchdogPhmActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management
Note	Definition of available actions to be applied to a watchdog. **Tags:** atp.Status=draft
Literal	*Description*
stopTrigger	**Tags:** atp.EnumerationValue=0
wrongTrigger	**Tags:** atp.EnumerationValue=1

Table 8.44: WatchdogPhmActionTypeEnum

8.3.6.4 Mode request action deployment

The Platform Health Management also supports the action to request a specific mode for a Process, Machine, or functionGroup.

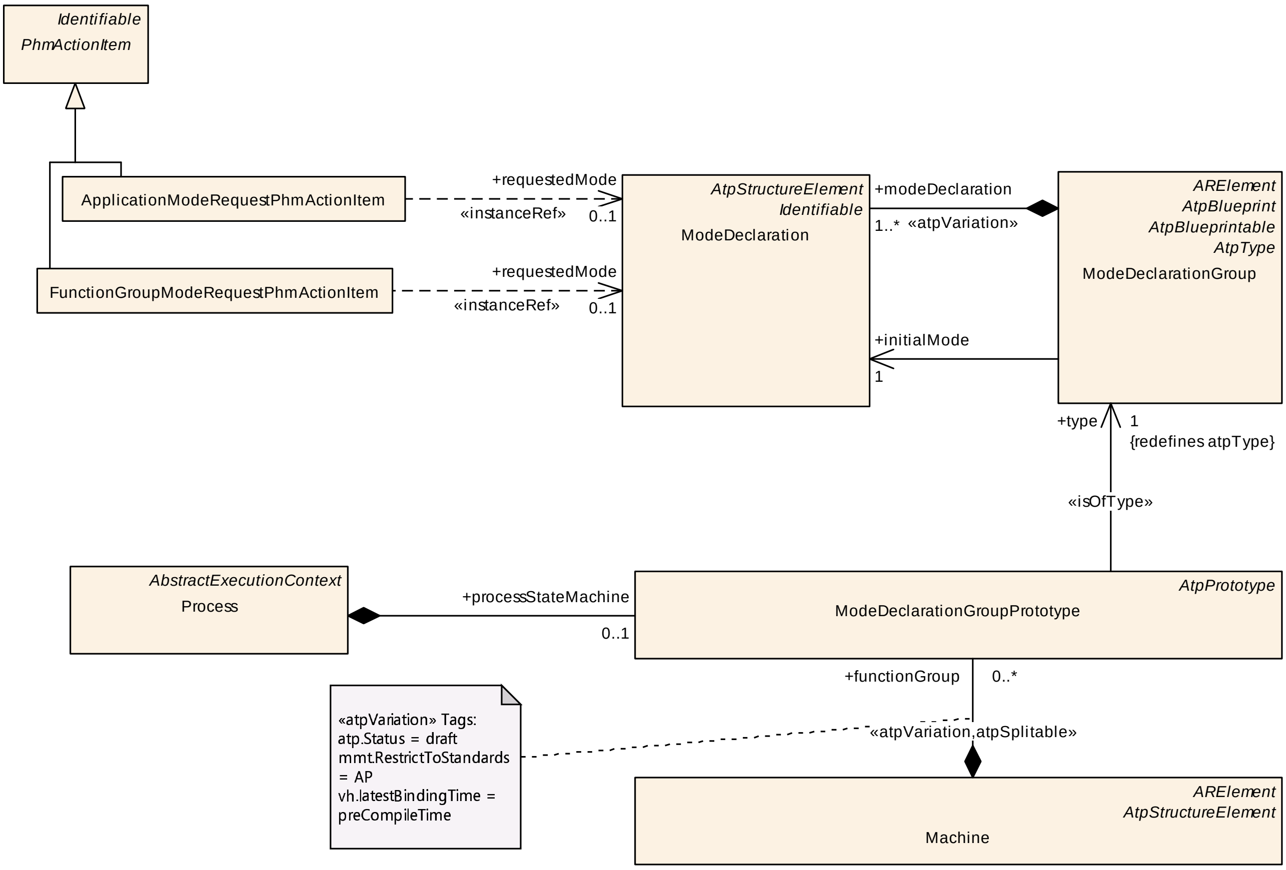


Figure 8.11: Modeling of mode request actions

Class	**ApplicationModeRequestPhmActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines the application mode request to be performed for one specific applicationMode Machine. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmActionItem*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
requestedMode	ModeDeclaration	0..1	iref	This reference defines the application mode request to be performed for one specific applicationModeMachine. **Tags:** atp.Status=draft

Table 8.45: ApplicationModeRequestPhmActionItem

Class	FunctionGroupModeRequestPhmActionItem			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines the function group mode request to be performed for one specific functionGroup. **Tags:** atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, PhmActionItem, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
requestedMode	ModeDeclaration	0..1	iref	This reference defines the function group mode request to be performed for one specific functionGroup. **Tags:** atp.Status=draft

Table 8.46: FunctionGroupModeRequestPhmActionItem

8.4 Time Synchronization Deployment

8.4.1 Overview

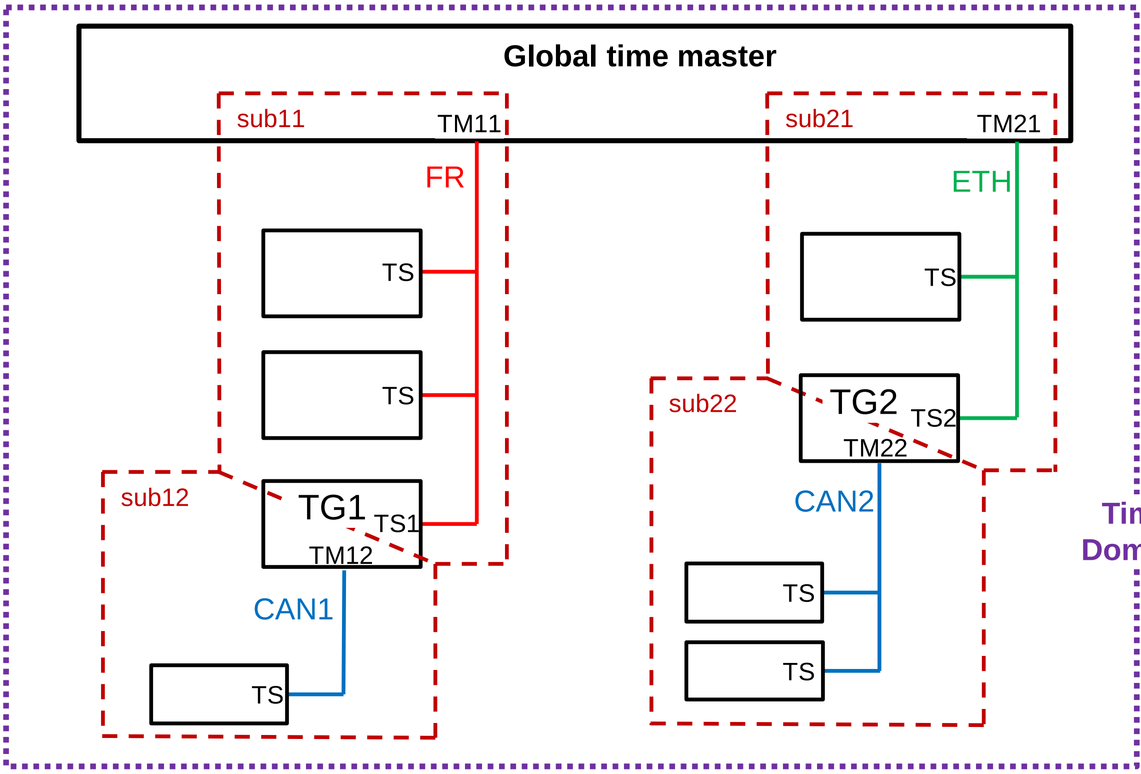
This chapter explains the configuration of the Time Synchronization functional cluster.

An adaptive AUTOSAR application can utilize several (synchronized) time base resources which are provided by the Time Synchronization functional cluster [11]. Time base resources can be *local* to the `Machine` or can be *synchronized* with a network `GlobalTimeDomain`.

The intended interaction of an adaptive AUTOSAR application with Time Synchronization is described in chapter 3.9.

Since an adaptive `Machine` is usually collaborating with other `Machine`s (adaptive) and ECUs (classic), special focus has been put on the vehicle wide definition of synchronized time. For a detailed specification please refer to the *Global Time Synchronization* chapter in the *System Template* [16].

Figure 8.12 provides an example system view on time domains and their transportation over diverse networks. In the scope of the *AUTOSAR adaptive platform* the focus is put on the Ethernet interaction with the rest of the system.



TS: GlobalTime slave
TM: GlobalTime master
TG: GlobalTime gateway

Figure 8.12: Example setup of Synchronized Global Time in AUTOSAR

8.4.2 Time Synchronization functional cluster configuration

The representation of the Time Synchronization functional cluster [11] within one specific `Machine` is defined by the `TimeSyncModuleInstantiation`. The `Machine` has the ability to define a set of `moduleInstantiation`s, where a specialization can be the `TimeSyncModuleInstantiation`.

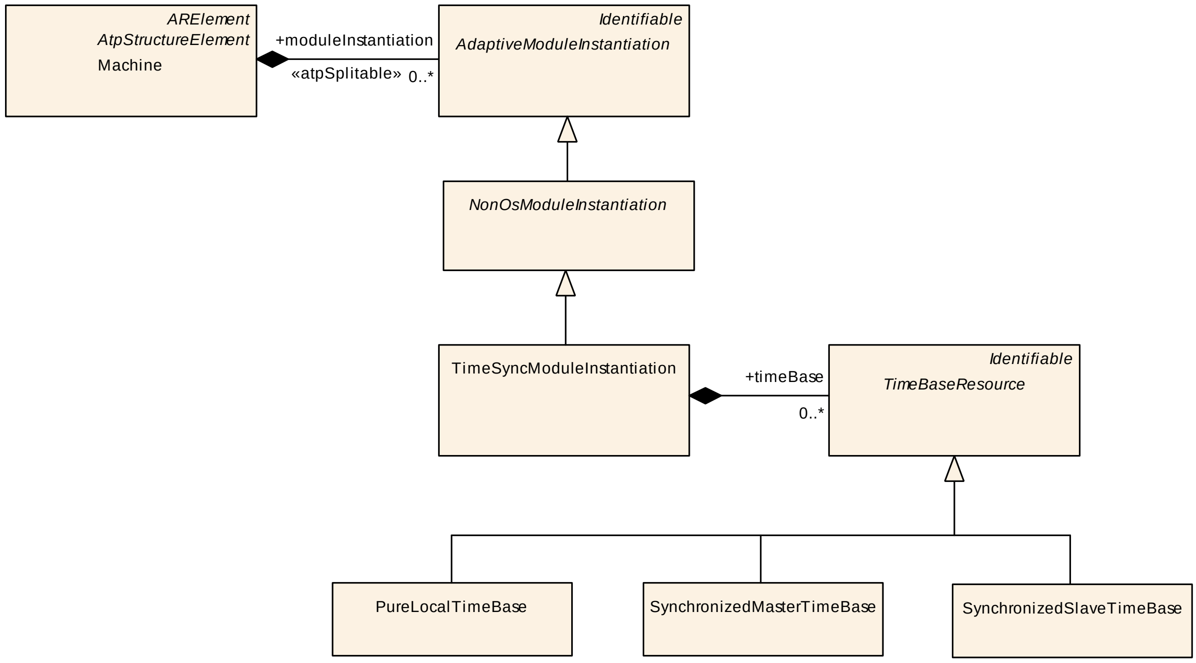


Figure 8.13: Modeling of TimeSyncModuleInstantiation

Class	**TimeSyncModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class defines the attributes for the Time Synchronization configuration on a specific machine. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
timeBase	TimeBaseResource	*	aggr	This aggregation defines the configured Time Bases for Time Synchronization. **Tags:** atp.Status=draft

Table 8.47: TimeSyncModuleInstantiation

8.4.3 Time Base

The TimeSyncModuleInstantiation represents the actual instance of the Time Synchronization functional cluster executed on a specific Machine. In the scope of the TimeSyncModuleInstantiation the Time Bases are defined.

[TPS_MANI_03539]{DRAFT} **Definition of Time Bases** ⌈ The meta-class TimeSyncModuleInstantiation has the ability to define a set of Time Bases of kind TimeBaseResource in the role timeBase. ⌋*(RS_MANI_00040)*

There are several sub types of TimeBaseResource which will be explained in the following sections.

Class	***TimeBaseResource*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the attributes of one Time Base for Time Synchronization. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PureLocalTimeBase, SynchronizedMasterTimeBase, SynchronizedSlaveTimeBase			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.48: TimeBaseResource

8.4.3.1 Pure local time base

A PureLocalTimeBase is a specialized TimeBaseResource with a Machine local scope only. So the time is neither received nor propagated over a network.

[TPS_MANI_03540]{DRAFT} **Definition of PureLocalTimeBase** ⌈ The meta-class PureLocalTimeBase defines a Time Base which is only available on the local Machine and is not *synchronized* over the network. ⌋*(RS_MANI_00040)*

Currently no further attributes are defined for the configuration of a PureLocalTimeBase.

Class	**PureLocalTimeBase**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents a Time Base which is maintained solely in the context of the local machine. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.49: PureLocalTimeBase

8.4.3.2 Synchronized time base

When configuring a synchronized time base many configuration aspects are already provided by the definition of the GlobalTimeDomain and are specified in the *System Template* [16] and associated to MachineDesign.

As for the configuration of the TimeSyncModuleInstantiation the usage of the SynchronizedMasterTimeBase respectively SynchronizedSlaveTimeBase defines the interaction with the GlobalTimeDomain.

[TPS_MANI_03541]{DRAFT} **Definition of SynchronizedSlaveTimeBase** ⌈ The meta-class SynchronizedSlaveTimeBase defines a Time Base which is synchronized with a time coming from the network. With the reference SynchronizedSlaveTimeBase.networkTimeSlave to a GlobalTimeSlave the relation to the system model is established. ⌋*(RS_MANI_00040)*

[TPS_MANI_03542]{DRAFT} **Definition of SynchronizedMasterTimeBase** ⌈ The meta-class SynchronizedMasterTimeBase defines a Time Base which is propagated to a time on the network. With the reference SynchronizedMasterTimeBase.networkTimeMaster to a GlobalTimeMaster the relation to the system model is established. ⌋*(RS_MANI_00040)*

Some aspects of the Synchronized Time Base for the *master* role are not available in the system model, those are provided with the TimeSyncCorrection.

[TPS_MANI_03543]{DRAFT} **Definition of time sync correction attributes** ⌈ The meta-class TimeSyncCorrection defines the attributes required to specify the time sync correction behavior of a SynchronizedMasterTimeBase. The SynchronizedMasterTimeBase aggregates the TimeSyncCorrection in the role timeSyncCorrection. ⌋*(RS_MANI_00040)*

The synchronized global time feature also supports the definition of *offset* time domains.

[TPS_MANI_03547]{DRAFT} **Definition of *offset* time domains** ⌈ A GlobalTimeDomain which has a offsetTimeDomain reference defined is considered an *offset* time domain. The reference source is the *offset* time domain. The reference *target* is the synchronized time domain. ⌋*(RS_MANI_00040)*

The *offset* time domain is applicable to GlobalTimeMaster (therefore also SynchronizedMasterTimeBase) and GlobalTimeSlave (therefore also SynchronizedSlaveTimeBase). *Offset* time domain is not applicable to PureLocalTimeBase.

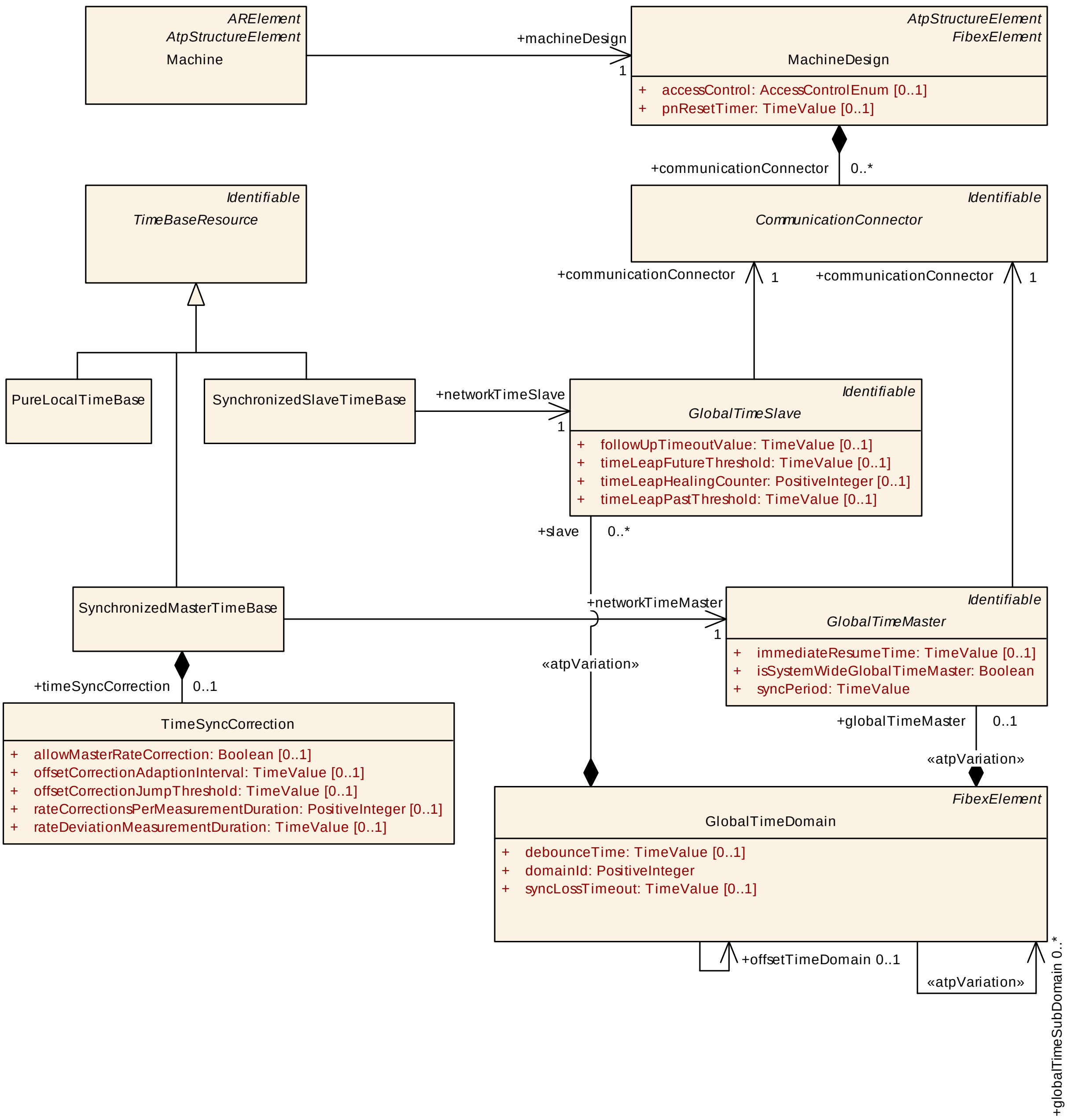


Figure 8.14: Modeling of synchronized time bases

Class	**SynchronizedSlaveTimeBase**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents a Synchronized Slave Time Base. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	**SynchronizedSlaveTimeBase**			
networkTime Slave	GlobalTimeSlave	1	ref	This reference defines the GlobalTimeSlave which is synchronized with this Time Base. **Tags:** atp.Status=draft

Table 8.50: SynchronizedSlaveTimeBase

Class	**SynchronizedMasterTimeBase**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents a Synchronized Master Time Base. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
networkTime Master	GlobalTimeMaster	1	ref	This reference defines the GlobalTimeMaster which is synchronized with this Time Base. **Tags:** atp.Status=draft
timeSync Correction	TimeSyncCorrection	0..1	aggr	This aggregation defines the attributes used for the correction of time synchronization. **Tags:** atp.Status=draft

Table 8.51: SynchronizedMasterTimeBase

Class	**TimeSyncCorrection**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the attributes used for the correction of time synchronization. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
allowMaster RateCorrection	Boolean	0..1	attr	Defines whether the rate correction value of a Time Base can be set by means of the method setRateCorrection(). false: rate correction cannot be set by method setRate Correction(). true: rate correction can be set by method setRate Correction(). **Tags:** atp.Status=draft
offsetCorrection AdaptionInterval	TimeValue	0..1	attr	Defines the interval during which the adaptive rate correction cancels out the rate and time deviation. Unit: seconds. **Tags:** atp.Status=draft
offsetCorrection JumpThreshold	TimeValue	0..1	attr	Threshold for the correction method. Deviations below this value will be corrected by a linear reduction over a defined timespan. Values equal and greater than this value will be corrected by immediately setting the correct time and rate in form of a jump. Unit: seconds. **Tags:** atp.Status=draft

▽

△

Class	**TimeSyncCorrection**			
rateCorrections Per Measurement Duration	PositiveInteger	0..1	attr	Number of simultaneous rate measurements to determine the current rate deviation. **Tags:** atp.Status=draft
rateDeviation Measurement Duration	TimeValue	0..1	attr	Time span used to calculate the rate deviation. Unit: seconds. **Tags:** atp.Status=draft

Table 8.52: TimeSyncCorrection

8.4.3.3 Ethernet synchronized time

As the *AUTOSAR adaptive platform* supports Ethernet as communication network also the time synchronization using Ethernet is supported.

In order to configure the behavior of the Ethernet time synchronization the specific sub-classes are used as shown in figure 8.15.

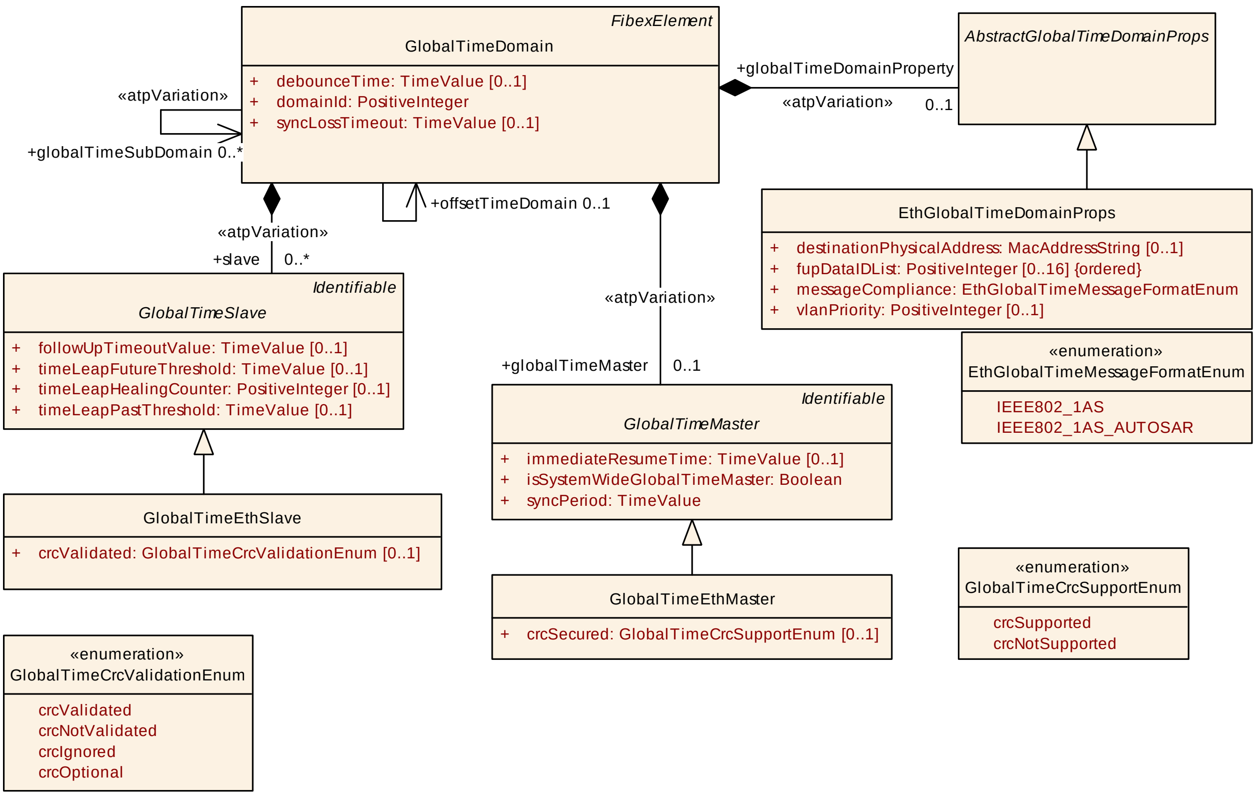


Figure 8.15: Modeling of Ethernet synchronized time

Class	**EthGlobalTimeDomainProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	Enables the definition of Ethernet Global Time specific properties.			
Base	*ARObject*, *AbstractGlobalTimeDomainProps*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
crcFlags	EthTSynCrcFlags	0..1	aggr	Defines the fields of the message which shall be taken into account for CRC calculation and verification.
destination Physical Address	MacAddressString	0..1	attr	Defines the MAC multicast address the Ethernet time sync messages are communicated on.
fupData IDList (ordered)	PositiveInteger	0..16	attr	The DataIDList for FUP messages to calculate CRC.
managed CouplingPort	EthGlobalTime ManagedCouplingPort	*	aggr	Collection of CouplingPorts which are managed in the scope of this Ethernet GlobalTimeDomain.
message Compliance	EthGlobalTimeMessage FormatEnum	1	attr	Defines the compliance of the Ethernet time sync messages to specific standards.
vlanPriority	PositiveInteger	0..1	attr	Defines which VLAN priority shall be assigned to a time sync message in case the message is sent using a VLAN tag.

Table 8.53: EthGlobalTimeDomainProps

Class	**GlobalTimeEthSlave**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	This represents the specialization of the GlobalTimeSlave for Ethernet communication.			
Base	*ARObject*, *GlobalTimeSlave*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
crcValidated	GlobalTimeCrc ValidationEnum	0..1	attr	Definition of whether or not validation of the CRC is supported.

Table 8.54: GlobalTimeEthSlave

Class	**GlobalTimeEthMaster**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	This represents the specialization of the GlobalTimeMaster for Ethernet communication.			
Base	*ARObject*, *GlobalTimeMaster*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
crcSecured	GlobalTimeCrcSupport Enum	0..1	attr	Definition of whether or not CRC is supported. This is only relevant for selected bus systems.
subTlvConfig	EthTSynSubTlvConfig	0..1	aggr	Defines the subTLV fields which shall be included in the time sync message.

Table 8.55: GlobalTimeEthMaster

8.4.4 Time Base to Port Prototype mapping

The `TimeBaseResource` definition of chapter 8.4.3 and the `RPortPrototype` typed by a sub-class of `TimeSynchronizationInterface` of chapter 3.9 have to be mapped to each other in order to define the binding of application software to the platform foundation software implementing the time synchronization.

[TPS_MANI_03548]{DRAFT} **Definition of `TimeSyncPortPrototypeToTimeBaseMapping`** ⌈ A `TimeSyncPortPrototypeToTimeBaseMapping` is used to define a mapping between a `TimeBaseResource` and a `RPortPrototype` typed by a sub-class of `TimeSynchronizationInterface` in the context of a `Process`. ⌋ *(RS_MANI_00040)*

The `TimeSyncPortPrototypeToTimeBaseMapping` takes the `Process` into account so that every instantiation of an `Executable` (and the resulting instantiation of all the `RPortPrototype`s typed by a sub-class of `TimeSynchronizationInterface`) can be mapped individually to `TimeBaseResource`s.

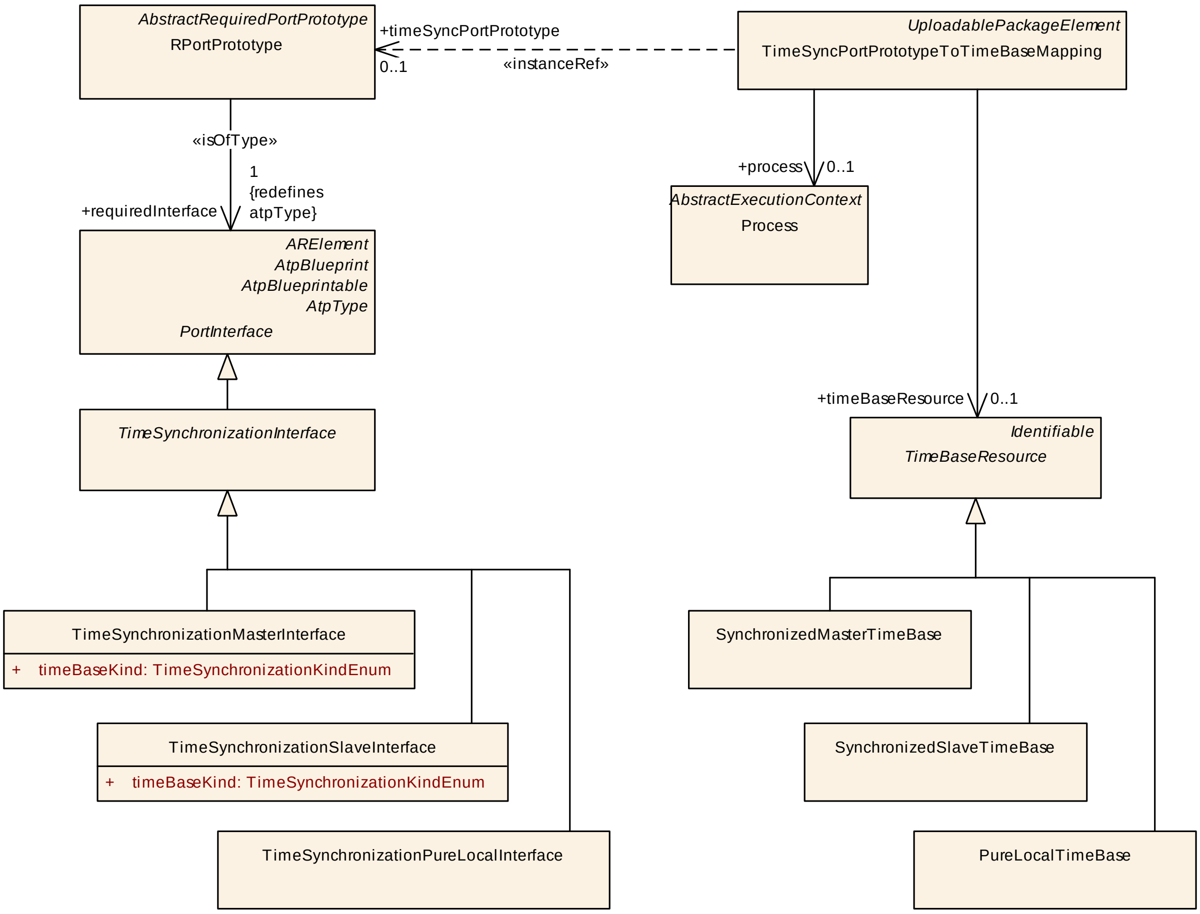


Figure 8.16: Modeling of `TimeSyncPortPrototypeToTimeBaseMapping`

Class	**TimeSyncPortPrototypeToTimeBaseMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync
Note	This meta-class provides the ability to map a RPortPrototype typed by a TimeSynchronizationInterface to a TimeBaseResource in the context of a Process. **Tags:** atp.Status=draft atp.recommendedPackage=TimeSyncPortPrototypeToTimeBaseMappings

▽

△

Class	TimeSyncPortPrototypeToTimeBaseMapping			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the context Process this mapping applies to. **Tags:** atp.Status=draft
timeBase Resource	TimeBaseResource	0..1	ref	Reference to the mapped TimeBaseResource. **Tags:** atp.Status=draft
timeSyncPort Prototype	RPortPrototype	0..1	iref	Instance reference to the mapped PortPrototype typed by a TimeSynchronizationInterface. **Tags:** atp.Status=draft

Table 8.56: TimeSyncPortPrototypeToTimeBaseMapping

The example shown in figure 8.17 illustrates the mapping of `RPortPrototype`s typed by one of the sub-classes of `TimeSynchronizationInterface` to actually configured `TimeBaseResource`s at the Time Sync Management.

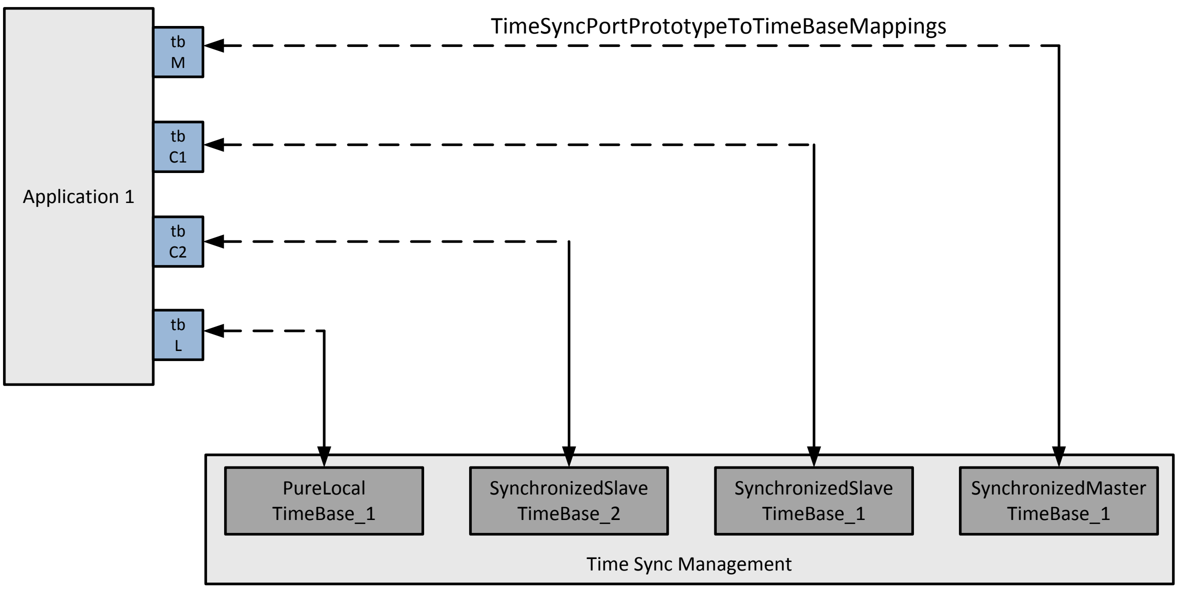


Figure 8.17: Example PortPrototype to TimeBase mapping

8.5 DoIP configuration

[TPS_MANI_03164]{DRAFT} **`Machine`-specific configuration settings for DoIP** ⌈
The `Machine`-specific configuration settings for DoIP are collected in `DoIpInstantiation`. ⌋*(RS_MANI_00023)*

Class	**DoIpInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the DoIP configuration on a specific machine. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject, AdaptiveModuleInstantiation, Identifiable, MultilanguageReferrable, NonOsModule Instantiation, Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
eid	PositiveUnlimitedInteger	0..1	attr	Configured EID (Entity ID) used for VehicleIdentification Request. If configured, take this value, if not configured use MAC address.
entityStatusMax ByteFieldUse	Boolean	1	attr	This attribute is used to distinguish the optional support of the Max data size element of a diagnostic entity status response.
gid	PositiveUnlimitedInteger	0..1	attr	Configured GID (Group ID) used for VehicleIdentification Request. If configured, take this value (and set "Further action required" byte to 0x00="No further action required"), if not configured use ServiceInterface DoIPGroupIdentification to retrieve GID and 'further action required' values.
gidInvalidity Pattern	PositiveInteger	1	attr	Specifies the Byte pattern that is used for response messages if no valid GID could be retrieved. Only the value '0' or '255' is allowed.
logicalAddress	PositiveInteger	1	attr	Describes the logical address of the DoIP entity, which is used for VehicleAnnouncement and RoutingActivation responses.
maxRequest Bytes	PositiveInteger	1	attr	Specifies the maximum allowed bytes of a DoIP message request without the DoIP header.
network Interface	DoIpNetwork Configuration	*	aggr	Network interface specific DoIP properties. **Tags:** atp.Status=draft
request Configuration	DoIpRequest Configuration	*	aggr	Request configuration that is used to determine whether an incoming DiagnosticMessage request needs to be interpreted as PHYSICAL or FUNCTIONAL. Any request with target address not within the configured target address range will be rejected. **Tags:** atp.Status=draft
vinInvalidity Pattern	PositiveInteger	1	attr	Specifies the Byte pattern that is used for response messages if no valid VIN could be retrieved. Only the value '0' or '255' is allowed.

Table 8.57: DoIpInstantiation

[constr_3425]{DRAFT} **Restriction of `DoIpInstantiation`s on a `Machine`** ⌈
Each `Machine` shall aggregate at most one `DoIpInstantiation` in the role `moduleInstantiation`. ⌋*()*

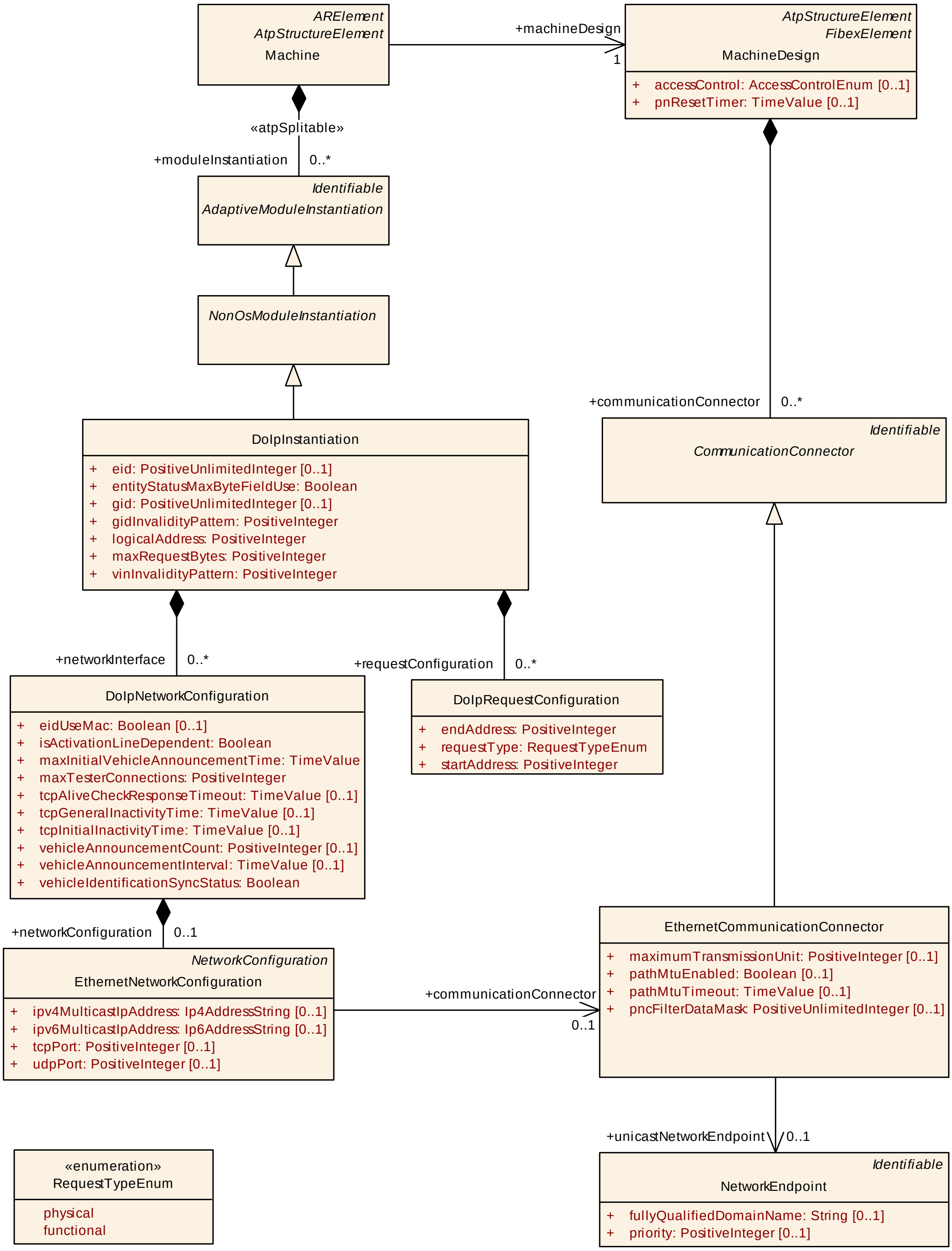


Figure 8.18: DoIP configuration

[constr_3495]{DRAFT} **Supported value range for attribute DoIpInstantiation.eid** ⌈ The supported value range of attribute DoIpInstantiation.eid is limited to the interval [0..281474976710655]. ⌋*()*

[constr_3496]{DRAFT} **Supported value range for attribute DoIpInstantiation.gid** ⌈ The supported value range of attribute DoIpInstantiation.gid is limited to the interval [0..281474976710655]. ⌋*()*

[constr_3497]{DRAFT} **Supported value range for attribute DoIpInstantiation.maxRequestBytes** ⌈ The supported value range of attribute DoIpInstantiation.maxRequestBytes is limited to the interval [0..4294967295]. ⌋*()*

[constr_3498]{DRAFT} **Supported value range for attribute DoIpInstantiation.logicalAddress** ⌈ The supported value range of attribute DoIpInstantiation.logicalAddress is limited to the interval [0..65535]. ⌋*()*

[TPS_MANI_03165]{DRAFT} **Network Interface configuration for DoIP** ⌈ The DoIpNetworkConfiguration contains all configuration settings that are specific for a configured network connection. The network connection is configured with the EthernetNetworkConfiguration that is aggregated by the DoIpNetworkConfiguration in the role networkConfiguration.

The attributes tcpPort and udpPort are used to configure the Transport Protocol (Udp or Tcp) and the used Port number. The IP Address is configured in the NetworkEndpoint that is referenced by the EthernetNetworkConfiguration via the EthernetCommunicationConnector. ⌋*(RS_MANI_00023)*

Class	**DoIpNetworkConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This element collects DoIP properties that are network interface specific. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
eidUseMac	Boolean	0..1	attr	This attribute defines whther the MAC of the network interface is used as eid. True: MAC is used False: eid needs to be configured manually by DoIp Instantiation.eid.
isActivationLine Dependent	Boolean	1	attr	This attribute defines whether the network interface • is started "on-demand" when an activation line is sensed or • is always available.
maxInitial Vehicle Announcement Time	TimeValue	1	attr	Upper bound for the time to wait in [s] for sending first vehicle anouncement message after IP address assignment. Represents parameter A_Do IP_Announce_Wait of ISO 13400-2:2012. The value of this timing shall be determined randomly in the closed interval [0..maxInitialVehicleAnnouncementTime].
maxTester Connections	PositiveInteger	1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.

▽

△

Class	DoIpNetworkConfiguration			
network Configuration	EthernetNetwork Configuration	0..1	aggr	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN. **Tags:** atp.Status=draft
tcpAliveCheck Response Timeout	TimeValue	0..1	attr	Timeout in [s] for waiting for a response to an Alive Check request before the connection is considered to be disconnected. Represents parameter T_TCP_AliveCheck of ISO 13400-2:2012.
tcpGeneral InactivityTime	TimeValue	0..1	attr	Timeout in [s] for maximum inactivity of a TCP socket connection before the DoIP module will close the according socket connection. Represents parameter T_TCP_General_Inactivity of ISO 13400-2:2012.
tcpInitial InactivityTime	TimeValue	0..1	attr	Timeout in [s] used for initial inactivity of a connected TCP socket connection directly after socket connection. Represents parameter T_TCP_Initial_Inactivity of ISO 13400-2:2012.
vehicle Announcement Count	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_Do IP_Announce_Num of ISO 13400-2:2012.
vehicle Announcement Interval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle anouncement messages. Represents parameter A_Do IP_Announce_Interval of ISO 13400-2:2012.
vehicle Identification SyncStatus	Boolean	1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/announcement.

Table 8.58: DoIpNetworkConfiguration

Please note that it is possible to define several networkInterfaces in a DoIpInstantiation. For each network connection individual configuration settings can be set with the attributes that are defined in the DoIpNetworkConfiguration element, e.g. it is possible to configure the vehicle announcement for different network connections in a different way.

[constr_5046]{DRAFT} **Usage of DoIpNetworkConfiguration.eidUseMac** ⌈ If DoIpInstantiation.eid is not configured the value of DoIpNetworkConfiguration.eidUseMac shall be set to true. ⌋*()*

[TPS_MANI_03218]{DRAFT} **Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration** ⌈ If no value for the attribute DoIpNetworkConfiguration.tcpInitialInactivityTime is defined then the default value of 2 seconds shall be assumed. ⌋*(RS_MANI_00023)*

[TPS_MANI_03219]{DRAFT} **Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration** ⌈ If no value for the attribute DoIpNetworkConfiguration.tcpGeneralInactivityTime is defined then the default value of 300 seconds shall be assumed. ⌋*(RS_MANI_00023)*

[TPS_MANI_03220]{DRAFT} **Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration** ⌈ If no value for the attribute DoIpNetworkConfiguration.vehicleAnnouncementCount is defined then the default value of 3 shall be assumed. ⌋*(RS_MANI_00023)*

[TPS_MANI_03221]{DRAFT} **Default value for the attribute vehicleAnnouncementInterval of meta-class DoIpNetworkConfiguration** ⌈ If no value for the attribute DoIpNetworkConfiguration.vehicleAnnouncementInterval is defined then the default value of 0,5 seconds shall be assumed. ⌋*(RS_MANI_00023)*

[TPS_MANI_03222]{DRAFT} **Default value for the attribute tcpAliveCheckResponseTimeout of meta-class DoIpNetworkConfiguration** ⌈ If no value for the attribute DoIpNetworkConfiguration.tcpAliveCheckResponseTimeout is defined then the default value of 0,5 seconds shall be assumed. ⌋*(RS_MANI_00023)*

During vehicle discovery the DoIP module responds by informing the tester about its own address, configured as the logicalAddress. The tester will approach the ECU under this UDS target address, thus the ECU should have a SoftwareCluster that is configured to respond to this UDS target address.

The list of available target addresses may or may not be obtainable from the SoftwareCluster with the logicalAddress.

In some cases, this SoftwareCluster may have the ability to inform the tester which other existing physical and/or logical addresses are available.

Class	**DoIpRequestConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class specifies a range of target addresses and its interpretation as either physical or functional request. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
endAddress	PositiveInteger	1	attr	End address for range of target-addresses (including this address).
requestType	RequestTypeEnum	1	attr	Determines the type of request.
startAddress	PositiveInteger	1	attr	Start address for range of target-addresses (including this address).

Table 8.59: DoIpRequestConfiguration

[constr_3492]{DRAFT} **DoIpInstantiation.logicalAddress shall be defined as member in the DoIpRequestConfiguration** ⌈ The DoIpInstantiation.logicalAddress shall be a member of the intervals of available physical addresses configured for the DoIpInstantiation in the requestConfiguration. ⌋*()*

On top of that, there is the expectation that the configured diagnosticAddresses of SoftwareClusters deployed to the Machine fit to the intervals defined in the context of the DoIpInstantiation in the requestConfiguration.

Please note that the DoIpRequestConfiguration corresponds to Table 39 that is defined in ISO-13400-2 [21].

[constr_3499]{DRAFT} **Supported value range for attribute DoIpRequest-Configuration.startAddress** ⌈ The supported value range of attribute DoIpRequestConfiguration.startAddress is limited to the interval [0..65535]. ⌋*()*

[constr_5000]{DRAFT} **Supported value range for attribute DoIpRequestConfiguration.endAddress** ⌈ The supported value range of attribute DoIpRequest-Configuration.endAddress is limited to the interval [0..65535]. ⌋*()*

8.6 Log and Trace module configuration

[TPS_MANI_03162]{DRAFT} **Machine-specific configuration settings for the Log and Trace functional cluster** ⌈ The Machine-specific configuration settings for the Log and Trace functional cluster are collected in LogAndTraceInstantiation. ⌋*(RS_MANI_00023)*

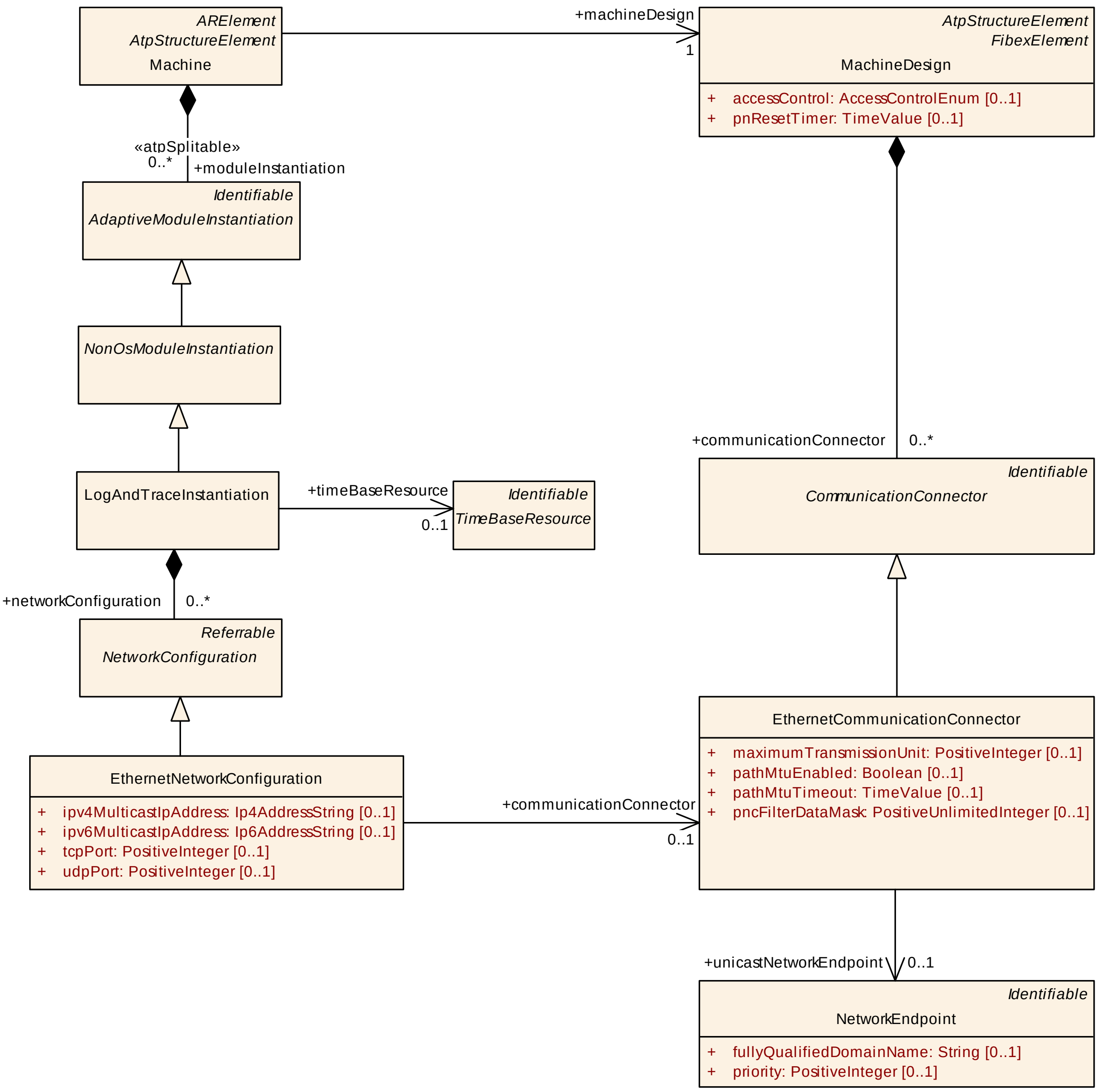


Figure 8.19: Log and Trace module configuration

Class	**LogAndTraceInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the Log&Trace configuration on a specific machine. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModule Instantiation*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
network Configuration	NetworkConfiguration	*	aggr	Network configuration for transmission of log & trace messages. **Tags:** atp.Status=draft

▽

△

Class	**LogAndTraceInstantiation**			
timeBase Resource	TimeBaseResource	0..1	ref	This reference is used to describe to which time base the the Log and Trace module has access. From the Time Base Resource the Log and Trace module gets the needed information to generate the time stamp. **Tags:** atp.Status=draft

Table 8.60: LogAndTraceInstantiation

[TPS_MANI_03163]{DRAFT} **Network configuration for Log and Trace messages** ⌈ The output channel on Ethernet for Log and Trace messages is configured with the EthernetNetworkConfiguration that is aggregated by the LogAndTraceInstantiation in the role networkConfiguration. The attributes tcpPort and udpPort are used to configure the Transport Protocol (Udp or Tcp) and the used Port number. The IP Address is configured in the NetworkEndpoint that is referenced by the EthernetNetworkConfiguration via the EthernetCommunicationConnector. ⌋*(RS_MANI_00023)*

8.7 Network Management configuration

[TPS_MANI_03166]{DRAFT} **Machine-specific configuration settings for NM module** ⌈ The Machine-specific configuration settings for Nm are collected in NmInstantiation. ⌋*(RS_MANI_00023)*

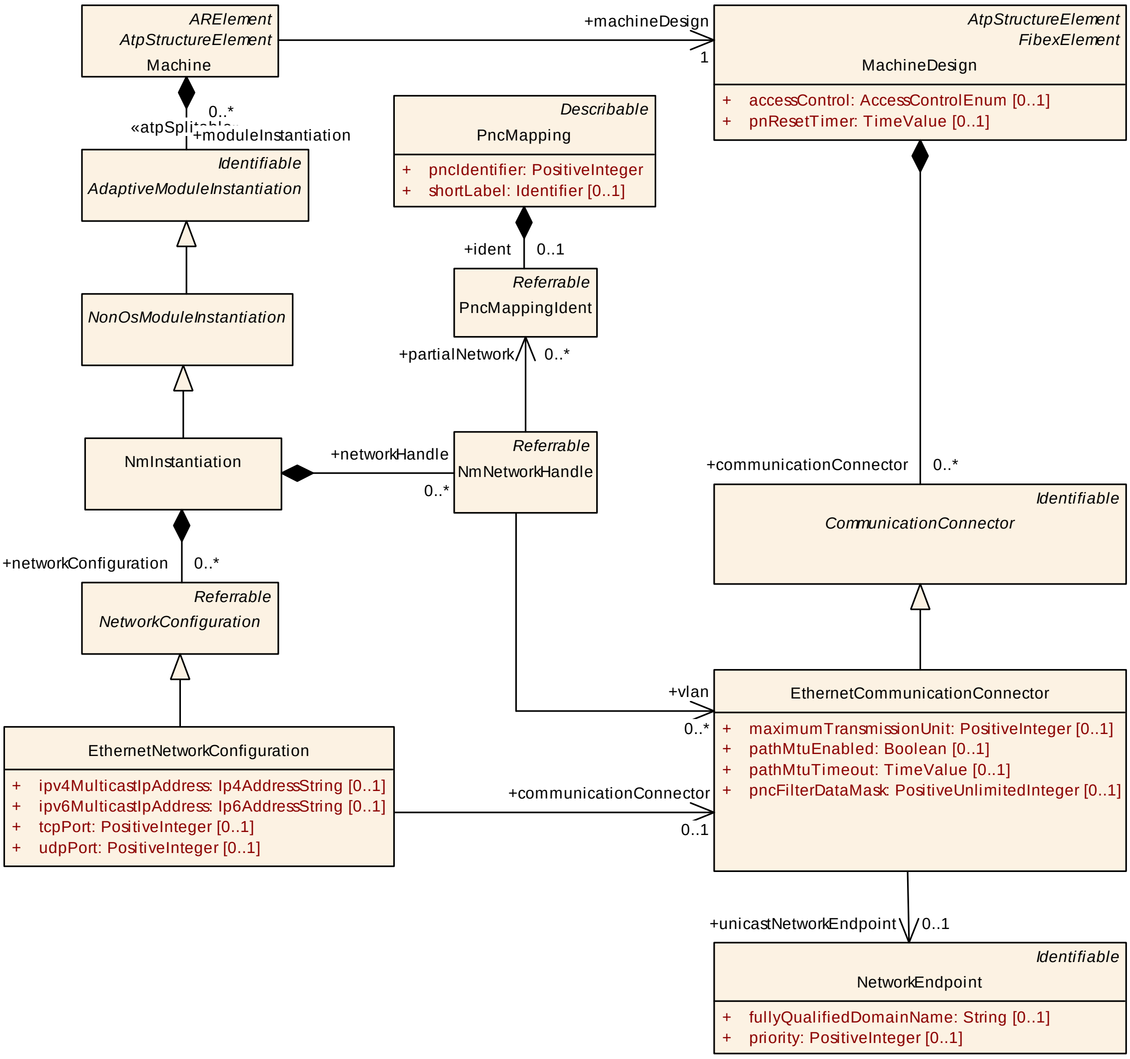


Figure 8.20: Network configuration for Nm

Class	**NmInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class defines the attributes for the Nm configuration on a specific machine. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
networkConfiguration	NetworkConfiguration	*	aggr	Network configuration for sending and receiving of Nm messages on the machine. **Tags:** atp.Status=draft

△

Class	NmInstantiation			
networkHandle	NmNetworkHandle	*	aggr	Supported NmNetworkHandles used to control Partial Network Clusters/VLANs. **Tags:** atp.Status=draft

Table 8.61: NmInstantiation

Class	NmNetworkHandle			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	Group of partialNetworks and/or VLANs that can be controlled collectively. **Tags:** atp.Status=draft			
Base	*ARObject*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
partialNetwork	PncMappingIdent	*	ref	Reference to a Partial Network that is included in the Nm NetworkHandle. **Tags:** atp.Status=draft
vlan	EthernetCommunication Connector	*	ref	Reference to a VLAN that is included in the NmNetwork Handle. **Tags:** atp.Status=draft

Table 8.62: NmNetworkHandle

[TPS_MANI_03226]{DRAFT} **Collection of partialNetworks and vlans in NmNetworkHandle** ⌈ The NmNetworkHandle element is used to describe a collection of partialNetworks and vlans that can be controlled collectively by the State Management. ⌋*(RS_MANI_00023)*

[TPS_MANI_03167]{DRAFT} **Network configuration for Nm** ⌈ The UDP multicast connection over which Network Management messages are transported is configured with the EthernetNetworkConfiguration that is aggregated by the NmInstantiation in the role networkConfiguration. The attribute udpPort is used to configure the port number. The IP Address is configured either by ipv4MulticastIpAddress or ipv6MulticastIpAddress. ⌋*(RS_MANI_00023)*

[constr_3414]{DRAFT} **Allowed usage of EthernetNetworkConfiguration attributes** ⌈ Table 8.63 shows EthernetNetworkConfiguration attributes that are allowed to be used to configure the network communication in the different platform modules. ⌋*()*

	Element		
EthernetNetworkConfiguration attributes	**Usage in NmInstantiation**	**Usage in DoIpInstantiation**	**Usage in LogAndTraceInstantiation**
tcpPort	N/A	Optional	Optional
udpPort	Mandatory	Optional	Optional
ipv4MulticastIpAddress	Optional	N/A	N/A
ipv6MulticastIpAddress	Optional	N/A	N/A
communicationConnector	Mandatory	Mandatory	Mandatory
udpNmCluster	Mandatory	N/A	N/A

Table 8.63: Allowed usage of EthernetNetworkConfiguration attributes

[constr_3419]{DRAFT} **Allowed usage of EthernetNetworkConfiguration attributes** ⌈ The EthernetNetworkConfiguration that is aggregated by NmInstantiation in the role networkConfiguration shall have either

- ipv4MulticastIpAddress or
- ipv6MulticastIpAddress.

⌋*()*

The UdpNmCluster with all included UdpNmNodes is described in the System design model. With the reference NmInstantiation.udpNmCluster the relation to the System design model is established.

Typically the System design model is provided by an OEM that defines the network configuration and provides all configuration settings that are relevant for a network management cluster to an integrator. The NM configuration options that will typically be set by an Integrator are collected in the NmInstantiation element. The Machine Manifest delivery to configure UdpNm consists of both, the NmInstantiation settings together with the referenced UdpNmCluster and UdpNmNode settings.

The NmConfig element is a wrapper that contains all network management specific configuration settings in the System model.

AUTOSAR Adaptive Network Management is based on periodic NM messages, which are received by all UdpNmNodes in the UdpNmCluster via multicast. Reception of NM packets indicates that sending UdpNmNodes want to keep the UdpNmCluster awake. If any node is ready to go to sleep mode, it stops sending NM messages, but as long as NM packets from other UdpNmNodes are received, it postpones transition to sleep mode.

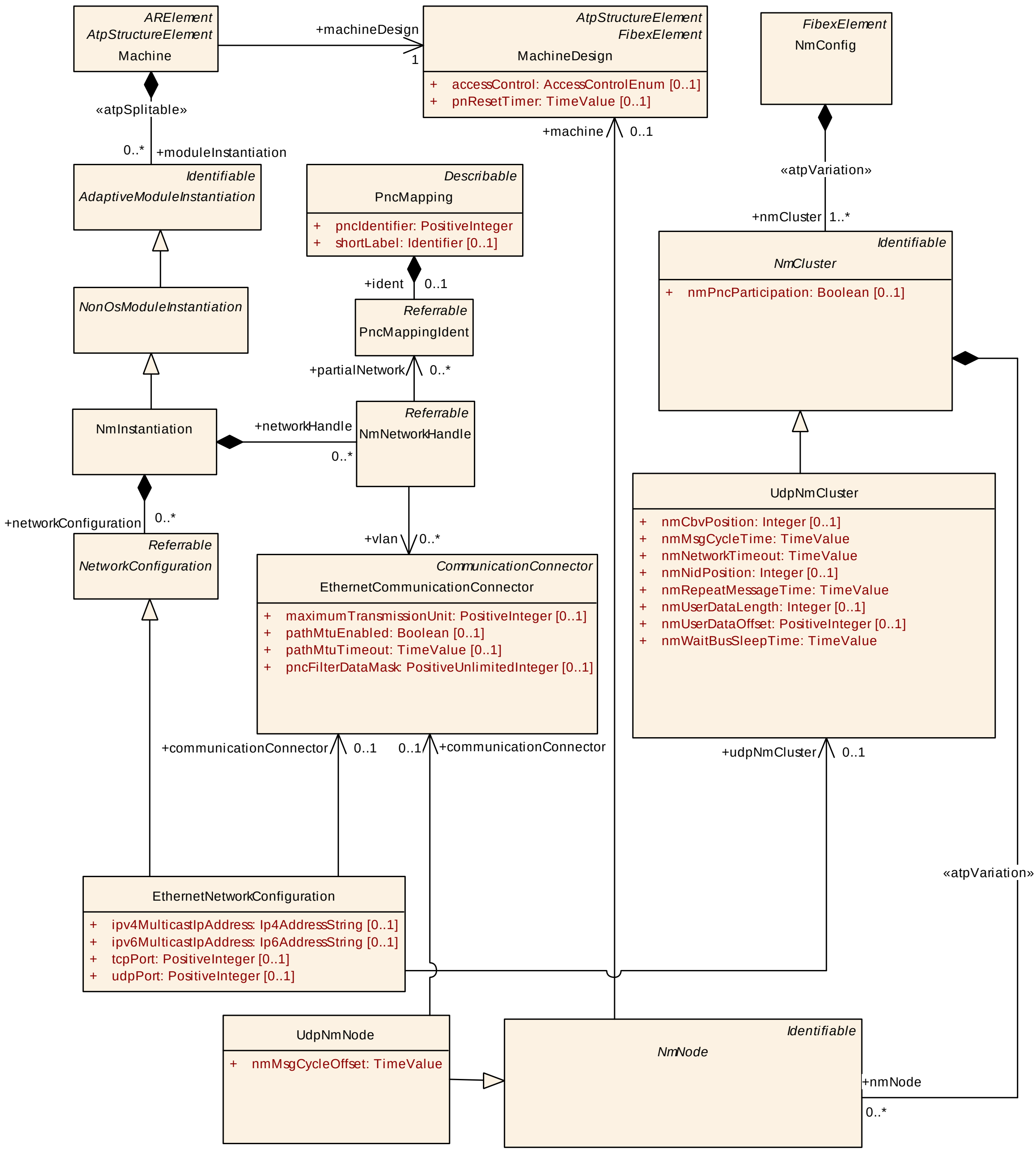


Figure 8.21: NM Cluster configuration

Class	NmConfig			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Contains the all configuration elements for AUTOSAR Nm. **Tags:** atp.ManifestKind=MachineManifest atp.recommendedPackage=NmConfigs			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
nmCluster	NmCluster	1..*	aggr	Collection of NM Clusters atpVariation: Derived, because cluster can be variable. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild

Table 8.64: NmConfig

Class	*NmCluster* (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Set of NM nodes coordinated with use of the NM algorithm. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanNmCluster, FlexrayNmCluster, J1939NmCluster, LinNmCluster, UdpNmCluster			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
communication Cluster	CommunicationCluster	1	ref	Asscociation to a CommunicationCluster in the topology description.
nmNode	NmNode	*	aggr	Collection of NmNodes of the NmCluster. atpVariation: Derived, because NmNode can be variable. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
nmPnc Participation	Boolean	0..1	attr	Defines whether this NmCluster contributes to the partial network mechanism.

Table 8.65: NmCluster

Class	UdpNmCluster			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Udp specific NmCluster attributes **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmCluster*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
nmCbvPosition	Integer	0..1	attr	Defines the position of the control bit vector within the Nm Pdu (Byte positon).
nmMsgCycle Time	TimeValue	1	attr	Period of a NmPdu in seconds. It determines the periodic rate in the periodic transmission mode with bus load reduction and is the basis for transmit scheduling in the periodic transmission mode without bus load reduction.

▽

△

Class	UdpNmCluster			
nmNetwork Timeout	TimeValue	1	attr	Network Timeout for NmPdus in seconds. It denotes the time how long the UdpNm shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place.
nmNidPosition	Integer	0..1	attr	Defines the byte position of the source node identifier within the NmPdu.
nmRepeat MessageTime	TimeValue	1	attr	Timeout for Repeat Message State in seconds. Defines the time how long the NM shall stay in the Repeat Message State.
nmUserData Length	Integer	0..1	attr	Defines the length in bytes of the user data contained in the Nm message. User data excludes the PN information.
nmUserData Offset	PositiveInteger	0..1	attr	Specifies the offset (in bytes) of the user data information in the NM message. User data excludes the PN information. **Tags:** atp.Status=draft
nmWaitBus SleepTime	TimeValue	1	attr	Timeout for bus calm down phase in seconds. It denotes the time how long the CanNm shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place.

Table 8.66: UdpNmCluster

Class	***NmNode*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	The linking of NmEcus to NmClusters is realized via the NmNodes. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanNmNode, FlexrayNmNode, J1939NmNode, UdpNmNode			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
machine	MachineDesign	0..1	ref	Reference to the machine that contains the NmNode. **Tags:** atp.Status=draft

Table 8.67: NmNode

Class	**UdpNmNode**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Udp specific NM Node attributes. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmNode*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
communication Connector	EthernetCommunication Connector	0..1	ref	Reference to the CommunicationConnector that represents the UdpNmNode in the topology description. **Tags:** atp.Status=draft
nmMsgCycle Offset	TimeValue	1	attr	Node specific time offset in the periodic transmission node. It determines the start delay of the transmission. Specified in seconds.

Table 8.68: UdpNmNode

8.8 Update and Configuration Management

[TPS_MANI_01226]{DRAFT} **Machine-specific configuration settings for the UCM module** ⌈ The Machine-specific configuration settings for Ucm are collected in meta-class UcmModuleInstantiation. ⌋*(RS_MANI_00023)*

Class	**UcmModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class represents the ability to define a definition of a UCM instantiation. **Tags:** atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
identifier	PositiveInteger	1	attr	This represents the identification of a UCM.

Table 8.69: UcmModuleInstantiation

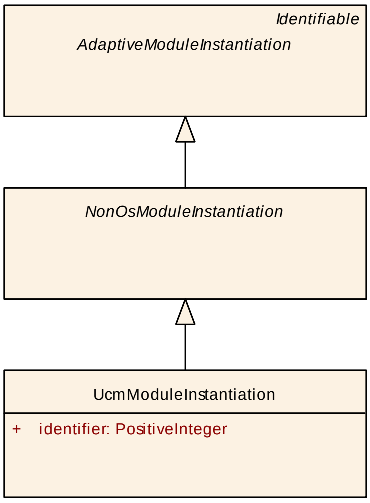


Figure 8.22: Modeling of UcmModuleInstantiation

[TPS_MANI_01227]{DRAFT} **Semantics of attribute UcmModuleInstantiation.identifier** ⌈ Attribute UcmModuleInstantiation.identifier shall be used to identify a specific Ucm on a specific Machine during a service discovery run by a master UCM or VUM. ⌋*(RS_MANI_00023)*

The usage of attribute UcmModuleInstantiation.identifier is documented in Figure 8.23. The master UCM or VUM acts as a client in a service discovery that is configured to search for *any* server.

The individual UCMs offer their service and then the master UCM as the client calls a specific method in the server's ServiceInterface to reveal the identifier of each server.

In the case of this example there are three slave UCMs with identifier set to 1, 2, and 3. The master UCM instantiates a proxy for each of the slave UCMs such that the value of the respective identifier can be retrieved from the proxy in order to be able to communicate with a specific slave UCM.

The master UCM or VUM can then instantiate proxies for each service offer and programmatically access the respective server going forward.

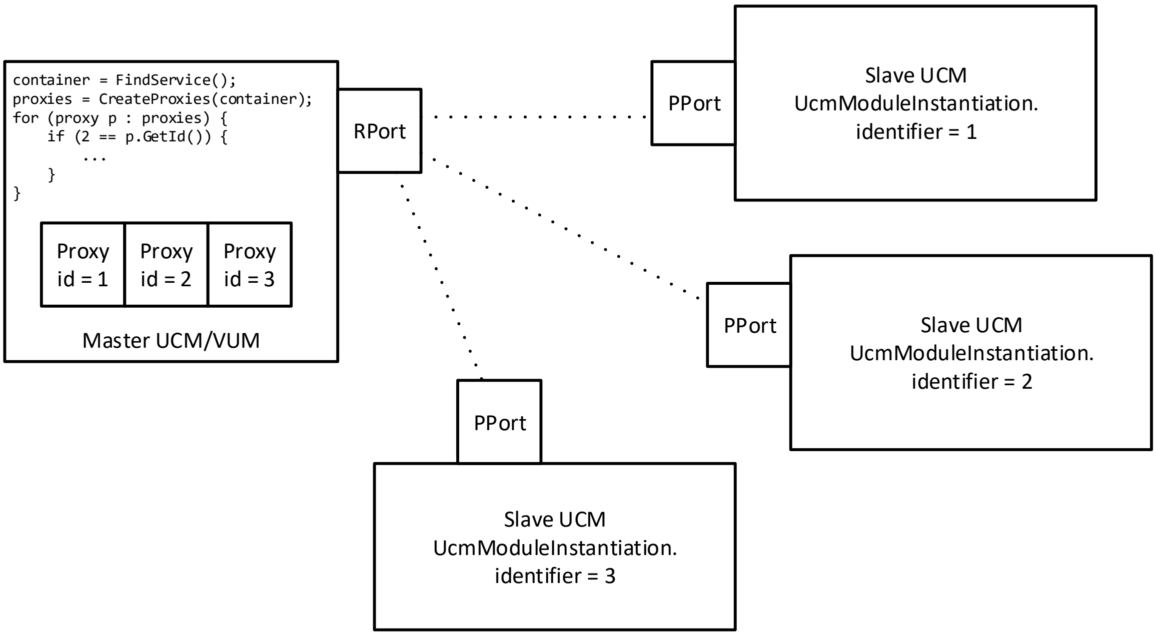


Figure 8.23: Identification of slave UCM modules by the master UCM

[constr_1691]{DRAFT} **UcmModuleInstantiation.identifier shall be unique**
⌈The value of attribute UcmModuleInstantiation.identifier shall be unique for each Machine in a given vehicle. ⌋*()*

8.9 IAM configuration

The definition of the deployment for the *Identity and Access Manager* represents the creation of actual grants, as opposed to the definition of grants on design level.

One important aspect of the modeling on deployment level is that it is not intended to include a large portion of the design model. The goal is to keep the deployment part as self-contained as possible.

While this approach represents a significant benefit for the size of deployment models it also creates some sort of a disconnect between design and deployment. In other words, the connection of the modeling of a specific Grant to the respective capabilities in the design model is not immediately obvious.

To mitigate this issue, AUTOSAR introduced the GrantDesign that in turn allows for the identification of the corresponding capability modeling. When loading the design model and deployment model together into a suitable tool it would still be possible to run an analysis in terms of completeness of the overall IAM configuration.

The enforcement of access restrictions is not mandatory for a Machine. Therefore, the existence of a Grant by itself is not sufficient to activate the IAM mechanisms.

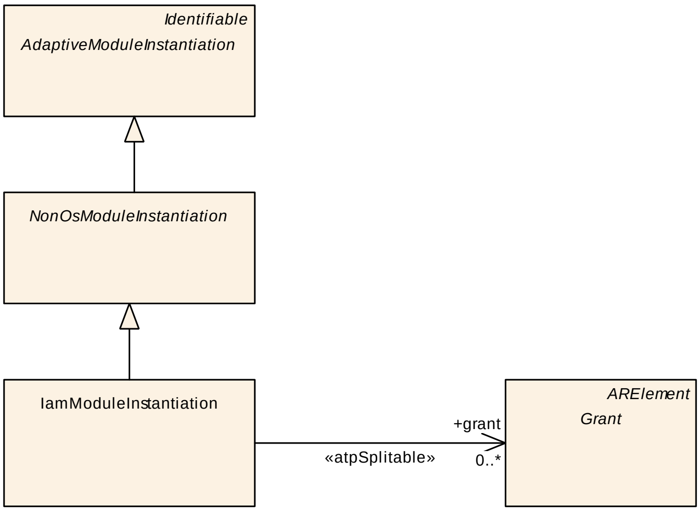


Figure 8.24: Modeling of the IamModuleInstantiation

[constr_1695]{DRAFT} **Semantics of a Grant depends on the existence of IamModuleInstantiation** ⌈ The existence of Grants shall only be enforced if in the context of the enclosing Machine an IamModuleInstantiation has been defined and is referencing the Grant. ⌋*()*

[TPS_MANI_01237]{DRAFT} **Semantics of meta-class ComFieldGrant** ⌈ Meta-class ComFieldGrant shall be used to award access to a given field (identified by means of the reference to meta-class ServiceFieldDeployment in the role serviceDeployment) in the context of a given AdaptivePlatformServiceInstance referenced in the role serviceInstance. ⌋*(RS_MANI_00060)*

In other words, if a given AdaptivePlatformServiceInstance and the respective ServiceFieldDeployment are not referenced from a ComFieldGrant and an IamModuleInstantiation exists **then this specific communication shall be suppressed**.

Class	**ComFieldGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.field. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	ComFieldGrant			
design	ComFieldGrantDesign	0..1	ref	This reference identifies the ComFieldGrantDesign that the enclosing ComFieldGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
role	FieldAccessEnum	1	attr	This attribute provides the ability to further specify the access to the ServiceInterface.field.
service Deployment	ServiceField Deployment	1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	1	ref	This reference identifies the applicable AdaptivePlatform ServiceInstance for which the grant applies. **Tags:** atp.Status=draft

Table 8.70: ComFieldGrant

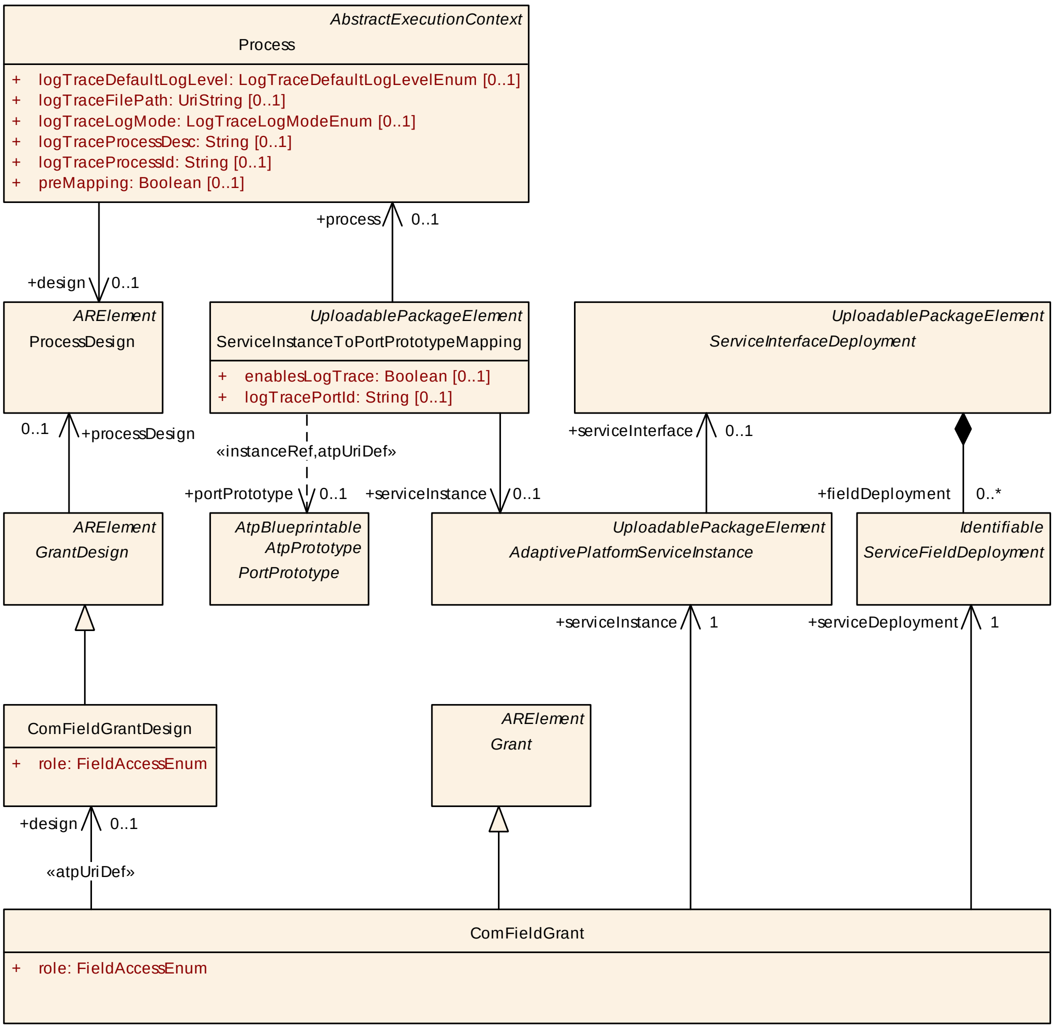


Figure 8.25: Modeling of the ComFieldGrant

[TPS_MANI_01238]{DRAFT} **Semantics of meta-class ComMethodGrant** ⌈ Meta-class ComMethodGrant shall be used to clear the call of a given method (identified by means of the reference to meta-class ServiceMethodDeployment in the role serviceDeployment) in the context of a given AdaptivePlatformServiceInstance referenced in the role serviceInstance. ⌋*(RS_MANI_00060)*

In other words, if a given AdaptivePlatformServiceInstance and the respective ServiceMethodDeployment are not referenced from a ComMethodGrant and an IamModuleInstantiation exists **then this specific communication shall be suppressed**.

Class	**ComMethodGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.method. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
design	ComMethodGrant Design	0..1	ref	This reference identifies the ComMethodGrantDesign that the enclosing ComMethodGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
service Deployment	ServiceMethod Deployment	1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	1	ref	This reference identifies the applicable AdaptivePlatform ServiceInstance for which the grant applies. **Tags:** atp.Status=draft

Table 8.71: ComMethodGrant

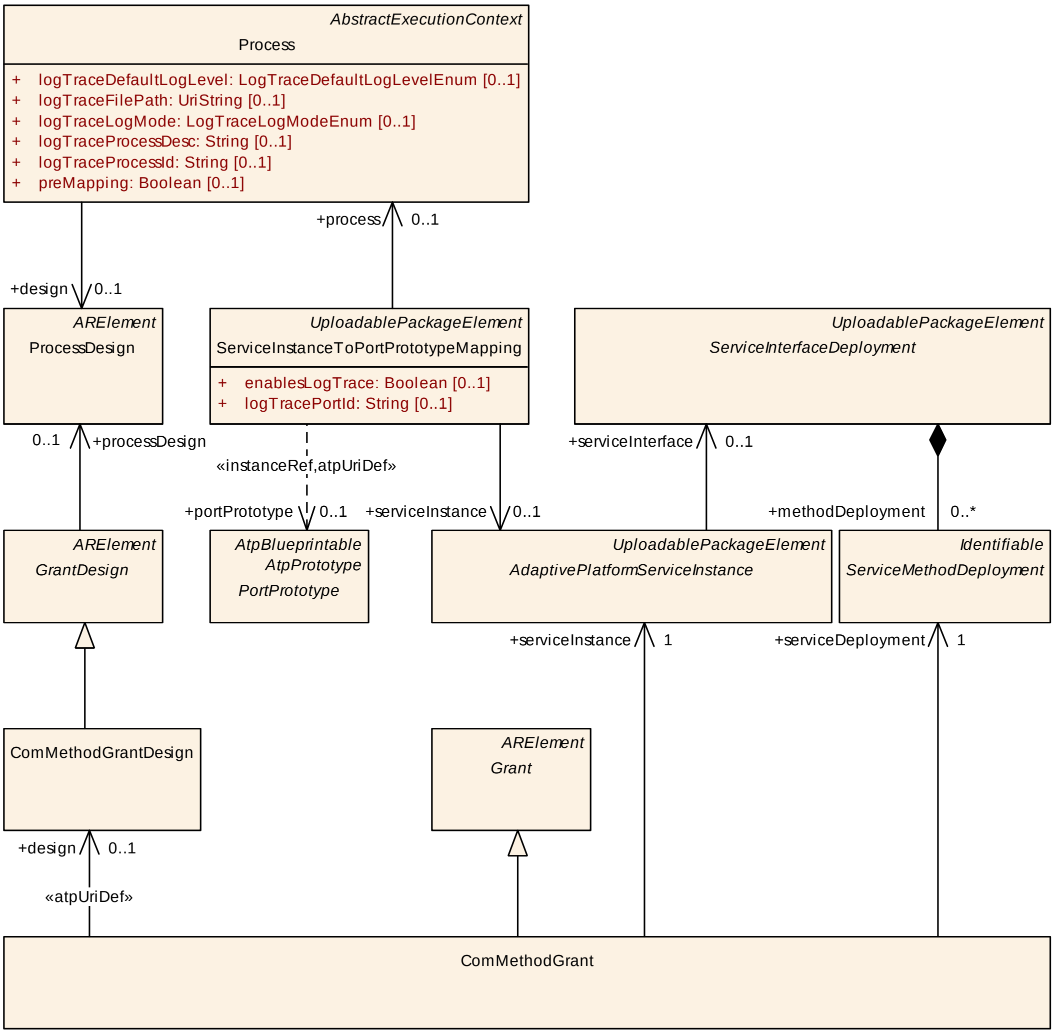


Figure 8.26: Modeling of the ComFieldGrant

[TPS_MANI_01239]{DRAFT} **Semantics of meta-class ComEventGrant** ⌈ Meta-class ComEventGrant shall be used to award access to a given event (identified by means of the reference to meta-class ServiceEventDeployment in the role serviceDeployment) in the context of a given AdaptivePlatformServiceInstance referenced in the role serviceInstance. ⌋*(RS_MANI_00060)*

In other words, if a given AdaptivePlatformServiceInstance and the respective ServiceMethodDeployment are not referenced from a ComEventGrant and an IamModuleInstantiation exists **then this specific communication shall be suppressed**.

Class	ComEventGrant			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.event. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
design	ComEventGrantDesign	0..1	ref	This reference identifies the ComEventGrantDesign that the enclosing ComEventGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
service Deployment	ServiceEvent Deployment	1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	1	ref	This reference identifies the applicable AdaptivePlatform ServiceInstance for which the grant applies. **Tags:** atp.Status=draft

Table 8.72: ComEventGrant

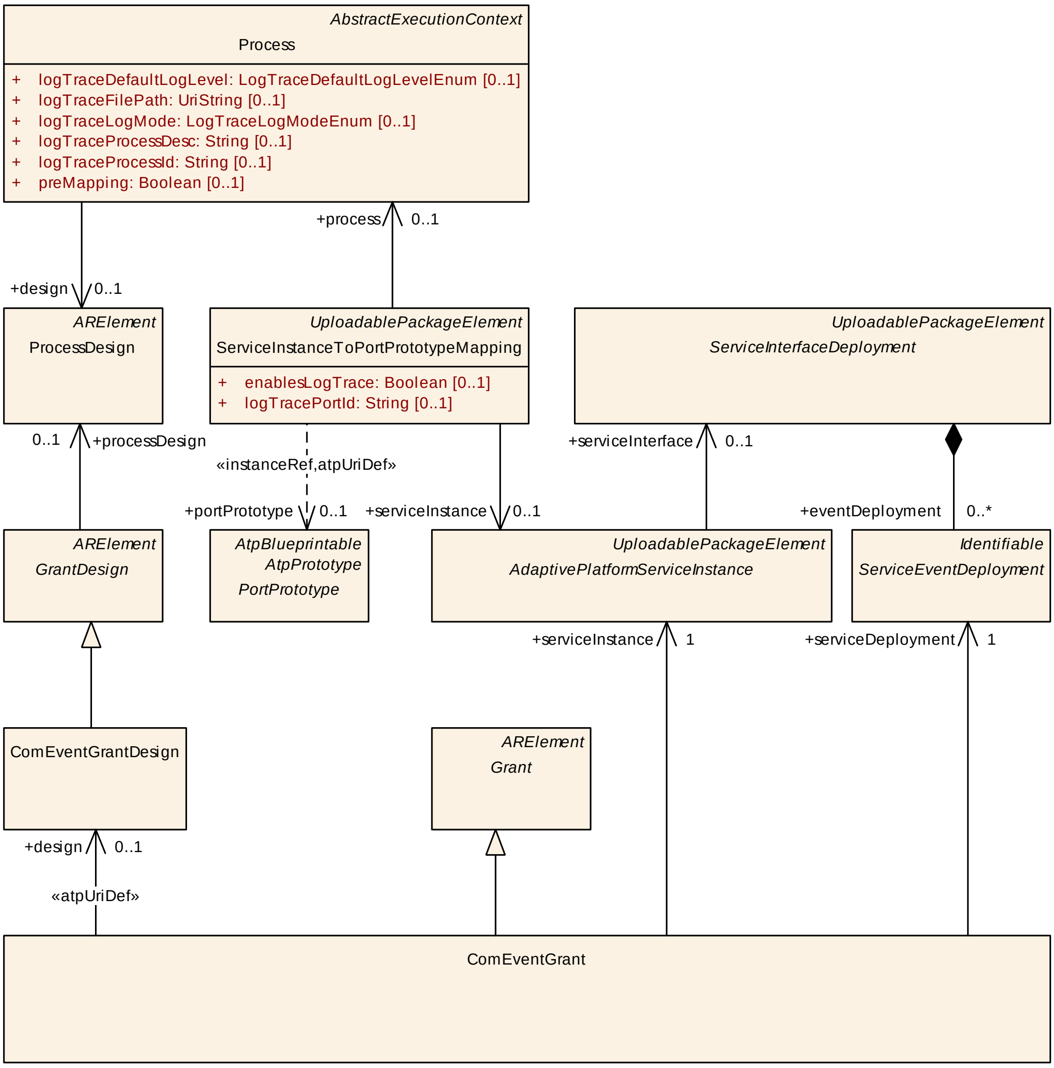


Figure 8.27: Modeling of the ComFieldGrant

The enforcement of service discovery rights is modeled by means of meta-classes ComOfferServiceGrant and ComFindServiceGrant.

[TPS_MANI_01240]{DRAFT} **Semantics of meta-class ComOfferServiceGrant**
⌈ Meta-class ComOfferServiceGrant shall be used to award the right to offer the referenced AdaptivePlatformServiceInstance. ⌋*(RS_MANI_00060)*

Class	**ComOfferServiceGrant**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement
Note	This meta-class represents the ability to grant the offering of a service. **Tags:** atp.Status=draft atp.recommendedPackage=Grants
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	ComOfferServiceGrant			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
design	ComOfferServiceGrant Design	0..1	ref	This reference identifies the ComOfferServiceGrant Design that the enclosing ComOfferServiceGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	1	ref	This reference identifies the AdaptivePlatformService Instances for which the grant applies. **Tags:** atp.Status=draft

Table 8.73: ComOfferServiceGrant

[TPS_MANI_01241]{DRAFT} **Semantics of meta-class ComFindServiceGrant** ⌈
Meta-class ComFindServiceGrant shall be used to award the right to start a find of the referenced AdaptivePlatformServiceInstance. ⌋*(RS_MANI_00060)*

Class	ComFindServiceGrant			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant the finding a service. **Tags:** atp.Status=draft atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
design	ComFindServiceGrant Design	0..1	ref	This reference identifies the ComFindServiceGrantDesign that the enclosing ComFindServiceGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	This reference identifies the AdaptivePlatformService Instances for which the grant applies. **Tags:** atp.Status=draft

Table 8.74: ComFindServiceGrant

9 Service Instance Manifest

9.1 Service Interface Deployment

The different meta-class specializations of `ServiceInterfaceDeployment` define a binding of a `ServiceInterface` to a middleware transport layer.

This chapter describes the usage of the `ServiceInterfaceDeployment` in different bindings that are supported by AUTOSAR.

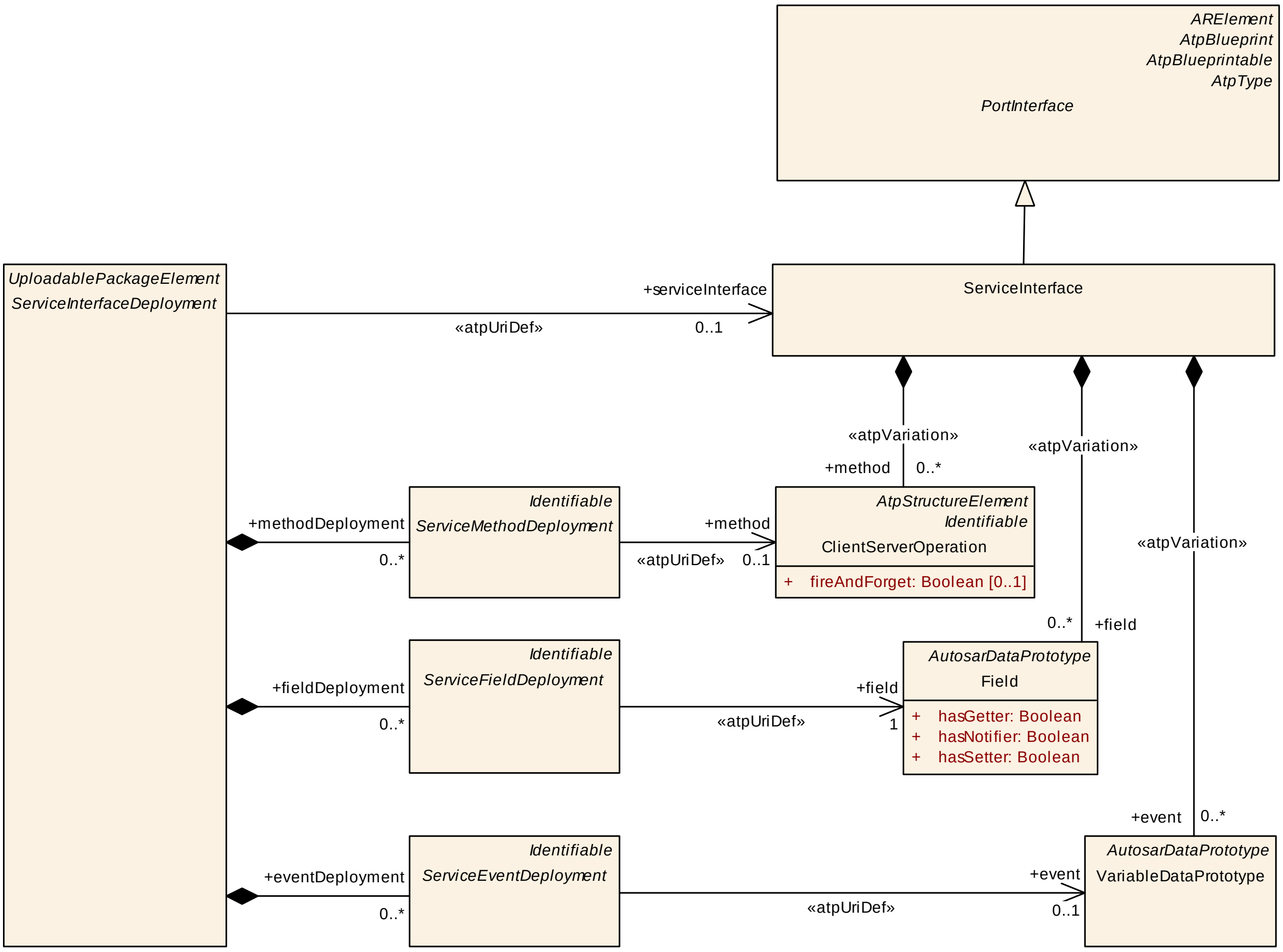


Figure 9.1: Deployment-related modeling of ServiceInterface

[TPS_MANI_03036]{DRAFT} **`ServiceInterface` deployment to a middleware transport layer** ⌈ The `ServiceInterfaceDeployment` meta-class provides the ability to map a `ServiceInterface` to a middleware transport layer that is represented by a concrete class that is derived from the abstract `ServiceInterfaceDeployment` meta-class. ⌋*(RS_MANI_00008)*

Class	***ServiceInterfaceDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	Middleware transport layer specific configuration settings for the ServiceInterface and all contained ServiceInterface elements. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	DdsServiceInterfaceDeployment, SignalBasedServiceInterfaceDeployment, SomeipServiceInterface Deployment, UserDefinedServiceInterfaceDeployment			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
event Deployment	ServiceEvent Deployment	*	aggr	Middleware transport layer specific configuration settings for an Event that is defined in the ServiceInterface. **Tags:** atp.Status=draft
fieldDeployment	ServiceField Deployment	*	aggr	Middleware transport layer specific configuration settings for a Field that is defined in the ServiceInterface. **Tags:** atp.Status=draft
method Deployment	ServiceMethod Deployment	*	aggr	Middleware transport layer specific configuration settings for a method that is defined in the ServiceInterface. **Tags:** atp.Status=draft
serviceInterface	ServiceInterface	0..1	ref	Reference to a ServiceInterface that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 9.1: ServiceInterfaceDeployment

[TPS_MANI_03037]{DRAFT} **Purpose of `ServiceMethodDeployment`** ⌈ The `ServiceMethodDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a `method` that is defined in the context of a `ServiceInterface`. ⌋*(RS_MANI_00008)*

[constr_3300]{DRAFT} **Allowed `ServiceMethodDeployment.method` references** ⌈ The `ClientServerOperation` that is referenced by `ServiceMethodDeployment` in the role `method` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceMethodDeployment`. ⌋*()*

[TPS_MANI_03038]{DRAFT} **Purpose of `ServiceEventDeployment`** ⌈ The `ServiceEventDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for an `event` that is defined in the context of a `ServiceInterface`. ⌋*(RS_MANI_00008)*

[constr_3301]{DRAFT} **Allowed `ServiceEventDeployment.event` references** ⌈ The `VariableDataPrototype` that is referenced by `ServiceEventDeployment` in the role `event` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceEventDeployment`. ⌋*()*

[TPS_MANI_03039]{DRAFT} **Purpose of ServiceFieldDeployment** ⌈ The ServiceFieldDeployment meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a field that is defined in the context of a ServiceInterface. ⌋*(RS_MANI_00008)*

[constr_3302]{DRAFT} **Allowed ServiceFieldDeployment.field references** ⌈ The Field that is referenced by ServiceFieldDeployment in the role field shall be defined in the context of a ServiceInterface that is referenced by the ServiceInterfaceDeployment in the role serviceInterface that contains the ServiceFieldDeployment. ⌋*()*

Class	***ServiceMethodDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of a Method to a middleware transport layer. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DdsMethodDeployment, SomeipMethodDeployment, UserDefinedMethodDeployment			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
method	ClientServerOperation	0..1	ref	Reference to a method that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 9.2: ServiceMethodDeployment

Class	***ServiceEventDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of an Event to a middleware transport layer. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DdsEventDeployment, SomeipEventDeployment, UserDefinedEventDeployment			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
event	VariableDataPrototype	0..1	ref	Reference to an Event that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 9.3: ServiceEventDeployment

Class	***ServiceFieldDeployment*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment

▽

△

Class	***ServiceFieldDeployment*** (abstract)			
Note	This abstract meta-class represents the ability to specify a deployment of a Field to a middleware transport layer. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable*			
Subclasses	DdsFieldDeployment, SomeipFieldDeployment, UserDefinedFieldDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
field	Field	1	ref	Reference to a Field that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 9.4: ServiceFieldDeployment

9.1.1 SOME/IP Service Interface Deployment

This chapter describes the SOME/IP deployment of a `ServiceInterface`.

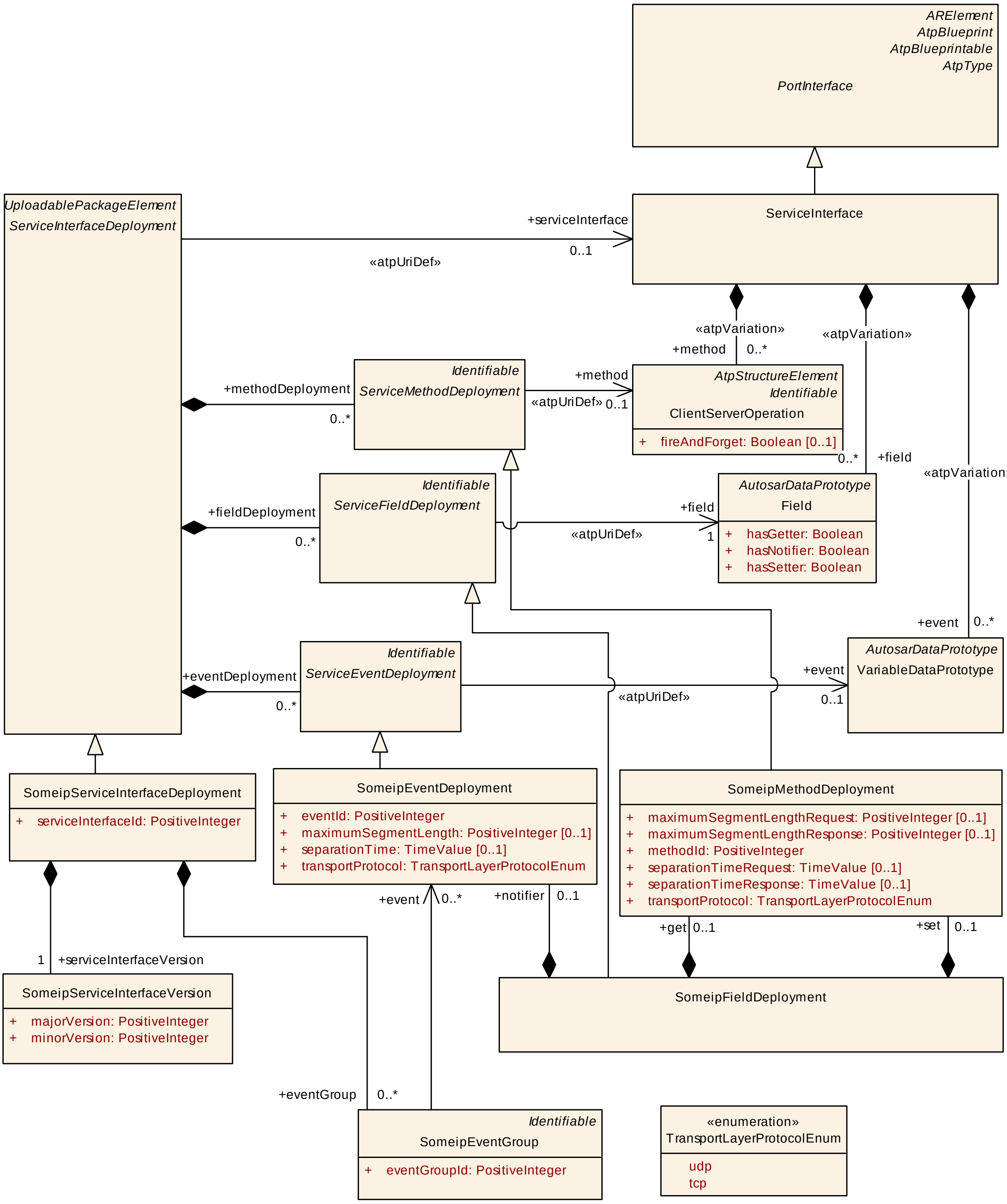


Figure 9.2: SOME/IP deployment of ServiceInterface

[TPS_MANI_03040]{DRAFT} **SOME/IP ServiceInterface binding** ⌈ The `SomeipServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` to SOME/IP and to assign a SOME/IP Service identifier to the `ServiceInterface` with the `serviceInterfaceId` attribute. ⌋*(RS_MANI_00024)*

The idea behind the SomeipServiceInterfaceDeployment is the definition of a common configuration set that is shared between the server that provides the ServiceInterface and all clients that are consuming the ServiceInterface. So it contains all relevant SOME/IP settings used for identification of the ServiceInterface and its content in messages on the network.

[constr_3410]{DRAFT} **Value range of SomeipServiceInterfaceDeployment.serviceInterfaceId** ⌈ The value of serviceInterfaceId shall be in the range of 0..65535. ⌋*()*

Please note that the SOME/IP MessageId that is 32 Bit long contains a 16 Bit serviceInterfaceId, a single bit that defines whether the message transports a method or an event and a 15 Bit eventId or methodId.

Please also consider [PRS_SOMEIPSD_00515] in [22] that defines special and reserved serviceInterfaceIds for SOME/IP and SOME/IP-SD.

[TPS_MANI_03041]{DRAFT} **Definition of SOME/IP EventGroups** ⌈ The SomeipServiceInterfaceDeployment.eventGroup allows to define SOME/IP *EventGroups* that are included in the SOME/IP Service and provide a logical grouping of events and notification events used for publish/subscribe handling. ⌋*(RS_MANI_00024)*

[constr_3304]{DRAFT} **Value of attribute SomeipEventGroup.eventGroupId shall be unique** ⌈ The value of attribute eventGroupId shall be unique in the context of the enclosing SomeipServiceInterfaceDeployment. ⌋*()*

[TPS_MANI_03042]{DRAFT} **Definition of SOME/IP Service Version** ⌈ The SomeipServiceInterfaceDeployment.serviceInterfaceVersion allows to define a major and a minor version for the SOME/IP Service. ⌋*(RS_MANI_00024)*

Class	**SomeipServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a ServiceInterface. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInterfaceDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	*	aggr	SOME/IP EventGroups that are defined within the SOME/IP ServiceClass. **Tags:** atp.Status=draft
serviceInterfaceId	PositiveInteger	1	attr	Unique Identifier that identifies the ServiceInterface in SOME/IP. This Identifier is sent as Service ID in SOME/IP Service Discovery messages.

▽

△

Class	SomeipServiceInterfaceDeployment			
serviceInterface Version	SomeipServiceInterface Version	1	aggr	The SOME/IP major and minor Version of the Service. **Tags:** atp.Status=draft

Table 9.5: SomeipServiceInterfaceDeployment

Class	**SomeipServiceInterfaceVersion**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe a version of a SOME/IP ServiceInterface. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
majorVersion	PositiveInteger	1	attr	Major Version of the ServiceInterface.
minorVersion	PositiveInteger	1	attr	Minor Version of the ServiceInterface.

Table 9.6: SomeipServiceInterfaceVersion

Class	**SomeipEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	Grouping of events and notification events inside a ServiceInterface in order to allow subscriptions. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
event	SomeipEvent Deployment	*	ref	Reference to an event that is part of the EventGroup. **Tags:** atp.Status=draft
eventGroupId	PositiveInteger	1	attr	Unique Identifier that identifies the EventGroup in SOME/IP. This Identifier is sent as Eventgroup ID in SOME/IP Service Discovery messages.

Table 9.7: SomeipEventGroup

[TPS_MANI_03043]{DRAFT} **SOME/IP `VariableDataPrototype` binding** ⌈ The `SomeipEventDeployment` meta-class provides the ability to bind a `VariableDataPrototype` to SOME/IP and to assign a SOME/IP Event identifier to the `event` with the `eventId` attribute. ⌋*(RS_MANI_00024)*

[constr_3305]{DRAFT} **Value of attribute `SomeipEventDeployment.eventId` shall be unique** ⌈ The value of `eventId` shall be unique in the in the context of the enclosing `SomeipServiceInterfaceDeployment` and shall also not overlap with any defined `methodId` used in the context of the enclosing `SomeipServiceInterfaceDeployment`. ⌋*()*

[constr_3408]{DRAFT} **Value range of `SomeipEventDeployment.eventId`** ⌈ The value of `eventId` shall be in the range of 0..32767. ⌋*()*

Please note that [PRS_SOMEIPSD_00517] in [22] defines special and reserved EVENT-IDs for SOME/IP and SOME/IP-SD that result in the eventId values of 0 and 32767.

[TPS_MANI_03050]{DRAFT} **Usage of SomeipEventDeployment.transportProtocol** ⌈The value of SomeipEventDeployment.transportProtocol defines over which Transport Layer Protocol the SomeipEventDeployment.event is provided. ⌋*(RS_MANI_00024)*

[constr_3307]{DRAFT} **SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances** ⌈ If SomeipEventDeployment.transportProtocol is set to udp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterface shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured udpPort. ⌋*()*

[constr_3308]{DRAFT} **SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances** ⌈ If SomeipEventDeployment.transportProtocol is set to tcp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterface shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured tcpPort. ⌋*()*

[TPS_MANI_03067]{DRAFT} **SOME/IP segmentation of udp SomeipEventDeployments** ⌈ If the maximumSegmentLength is set to a value and the data length is larger than maximumSegmentLength then SOME/IP shall segment the SomeipEventDeployment into several packets and transmit them over the network.

The sender shall wait the separationTime between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipEventDeployment. ⌋*(RS_MANI_00024)*

[constr_3351]{DRAFT} **SOME/IP segmentation allowed for udp SomeipEventDeployments** ⌈ Attribute SomeipEventDeployment.maximumSegmentLength shall only be used if the value of attribute SomeipEventDeployment.transportProtocol is set to udp. ⌋*()*

Class	**SomeipEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for an Event. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceEventDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	**SomeipEventDeployment**			
eventId	PositiveInteger	1	attr	Unique Identifier within a ServiceInterface that identifies the Event in SOME/IP. This Identifier is sent as part of the Message ID in SOME/IP messages.
maximum SegmentLength	PositiveInteger	0..1	attr	This attribute describes the length in bytes of the SOME/IP segment. This includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLength then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
separationTime	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments.
transport Protocol	TransportLayerProtocol Enum	1	attr	This attribute defines over which Transport Layer Protocol this event is intended to be sent.

Table 9.8: SomeipEventDeployment

[TPS_MANI_03044]{DRAFT} **SOME/IP ClientServerOperation binding** ⌈ The SomeipMethodDeployment meta-class provides the ability to bind a ClientServerOperation to SOME/IP and to assign a SOME/IP Method identifier to the method with the methodId attribute. ⌋*(RS_MANI_00024)*

[constr_3306]{DRAFT} **Value of attribute methodId shall be unique per SomeipServiceInterfaceDeployment** ⌈ The value of methodId shall be unique in the in the context of the enclosing SomeipServiceInterfaceDeployment and shall also not overlap with any defined eventId used in the context of the enclosing SomeipServiceInterfaceDeployment. ⌋*()*

[constr_3409]{DRAFT} **Value range of SomeipMethodDeployment.methodId** ⌈ The value of methodId shall be in the range of 0..32767. ⌋*()*

Please note that [PRS_SOMEIPSD_00517] in [22] defines special and reserved METHOD-IDs for SOME/IP and SOME/IP-SD that result in the methodId values of 0 and 32767.

[TPS_MANI_03051]{DRAFT} **Usage of SomeipMethodDeployment.transportProtocol** ⌈ The value of SomeipMethodDeployment.transportProtocol defines over which Transport Layer Protocol this method is provided. ⌋*(RS_MANI_00024)*

[constr_3309]{DRAFT} **SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances** ⌈ If SomeipMethodDeployment.transportProtocol is set to udp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterface shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured udpPort. ⌋*()*

[constr_3310]{DRAFT} **SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances** ⌈ If SomeipMethodDeployment.transportProtocol is set to tcp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterface shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured tcpPort. ⌋*()*

[TPS_MANI_03068]{DRAFT} **SOME/IP segmentation of SomeipMethodDeployment Calls** ⌈ If the maximumSegmentLengthRequest is set to a value and the data length is larger than maximumSegmentLengthRequest then SOME/IP shall segment the SomeipMethodDeployment Call-Message into several packets and transmit them over the network.

The sender shall wait the separationTimeRequest between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipMethodDeployment Call-Message. ⌋ *(RS_MANI_00024)*

[TPS_MANI_03069]{DRAFT} **SOME/IP segmentation of SomeipMethodDeployment Responses** ⌈ If the maximumSegmentLengthResponse is set to a value and the data length is larger than maximumSegmentLengthResponse then SOME/IP shall segment the SomeipMethodDeployment Response-Message into several packets and transmit them over the network.

The sender shall wait the separationTimeResponse between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipMethodDeployment Response-Message. ⌋*(RS_MANI_00024)*

[constr_3352]{DRAFT} **SOME/IP segmentation allowed for udp SomeipMethodDeployments** ⌈ SomeipMethodDeployment.maximumSegmentLengthRequest and SomeipMethodDeployment.maximumSegmentLengthResponse shall only be used if SomeipMethodDeployment.transportProtocol is set to udp. ⌋*()*

Class	**SomeipMethodDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a Method. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceMethodDeployment*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	**SomeipMethodDeployment**			
maximumSegmentLengthRequest	PositiveInteger	0..1	attr	This attribute describes the length in bytes of one SOME/IP segment into which the Method Call Message will be divided. This length field includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLengthRequest then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
maximumSegmentLengthResponse	PositiveInteger	0..1	attr	This attribute describes the length in bytes of one SOME/IP segment into which the Method Return Message will be divided. This length field includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLengthResponse then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
methodId	PositiveInteger	1	attr	Unique Identifier within a ServiceInterface that identifies the Method in SOME/IP. This Identifier is sent as part of the Message ID in SOME/IP messages.
separationTimeRequest	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments into which the Method Call Message will be divided.
separationTimeResponse	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments into which the Method Return Message will be divided.
transportProtocol	TransportLayerProtocolEnum	1	attr	This attribute defines over which Transport Layer Protocol this method is intended to be sent.

Table 9.9: SomeipMethodDeployment

Class	**SomeipServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map SomeipServiceInstances to a CommunicationConnector of a Machine. In this step the network configuration (IP Address, Transport Protocol, Port Number) for the ServiceInstance is defined. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventMulticastUdpPort	PositiveInteger	0..1	attr	UdpPort configuration that is used for Event communication in the IP-Multicast case. During SOME/IP Service Discovery: Send in the SD-SubscribeEventGroupAck Message to client (answer to SD-SubscribeEventGroup). ▽

▽

△

Class	SomeipServiceInstanceToMachineMapping			
				△ Event: This is the destination-port where the server sends the multicast event messages if the multicastThreshold of the corresponding SomeipProvidedEventGroup is exceeded.
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address that is transmitted in the EventGroupSubscribeAck message for all available EventGroups that are available in the ProvidedSomeipServiceInstance.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address that is transmitted in the EventGroupSubscribeAck message for all available EventGroups that are available in the ProvidedSomeipServiceInstance.
tcpPort	PositiveInteger	0..1	attr	TcpPort configuration that is used for Method and Event communication in IP-Unicast case. During SOME/IP Service Discovery: PortNumber that is sent in the SD-Offer Message to client (answer on SD-find) or clients (SD-offer). Method: This is the destination-port where the server accepts the method call messages (from the clients). This is the source-port where the server sends the method response messages (to the client). Event: This is the event source-port where the server sends the event messages to the subscribed clients in IP-Unicast case.
udpCollection BufferSize Threshold	PositiveInteger	0..1	attr	Specifies the amount of data in bytes that shall be buffered for data transmission over the udp connection specified by this SomeipServiceInstanceToMachine Mapping in case data collection is enabled.
udpPort	PositiveInteger	0..1	attr	UdpPort configuration that is used for Method and Event communication in IP-Unicast case. During SOME/IP Service Discovery: PortNumber that is sent in the SD-Offer Message to client (answer on SD-find) or clients (SD-offer). Method: This is the destination-port where the server accepts the method call messages (from the clients). This is the source-port where the server sends the method response messages (to the client). Event: This is the event source-port where the server sends the event messages to the subscribed clients in IP-Unicast case.

Table 9.10: SomeipServiceInstanceToMachineMapping

[TPS_MANI_03057]{DRAFT} **SOME/IP `Field` binding** ⌈ The `SomeipFieldDeployment` meta-class provides the ability to bind a `Field` to SOME/IP.

If the `Field` contains a notifier (`hasNotifier` = true) it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipFieldDeployment.notifier.eventId`.

If the `Field` contains a getter method (`hasGetter` = true) it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipFieldDeployment.get.methodId`.

If the Field contains a setter method (hasSetter = true) it is possible to assign a SOME/IP notifier identifier to the field by setting the value of attribute SomeipFieldDeployment.set.methodId ⌋*(RS_MANI_00024)*

Please note that each methodId and each eventId of a SomeipFieldDeployment shall be unique in the context of a ServiceInterface as defined in [constr_3306] and [constr_3305].

Class	**SomeipFieldDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a Field. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceFieldDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
get	SomeipMethodDeployment	0..1	aggr	This aggregation represents the setting of the get method. **Tags:** atp.Status=draft
notifier	SomeipEventDeployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:** atp.Status=draft
set	SomeipMethodDeployment	0..1	aggr	This aggregation represents the settings of the set method **Tags:** atp.Status=draft

Table 9.11: SomeipFieldDeployment

[constr_3362]{DRAFT} **SomeipEventDeployments aggregated by a SomeipFieldDeployment** ⌈A SomeipEventDeployment that is aggregated by a SomeipFieldDeployment in the role notifier shall not reference a VariableDataPrototype in the role event. ⌋*()*

[constr_3363]{DRAFT} **SomeipMethodDeployments aggregated by a SomeipFieldDeployment** ⌈ A SomeipMethodDeployment that is aggregated by a SomeipFieldDeployment in the role get or set shall not reference a ClientServerOperation in the role method. ⌋*()*

9.1.2 DDS Service Interface Deployment

This chapter describes the DDS [23] deployment of a ServiceInterface.

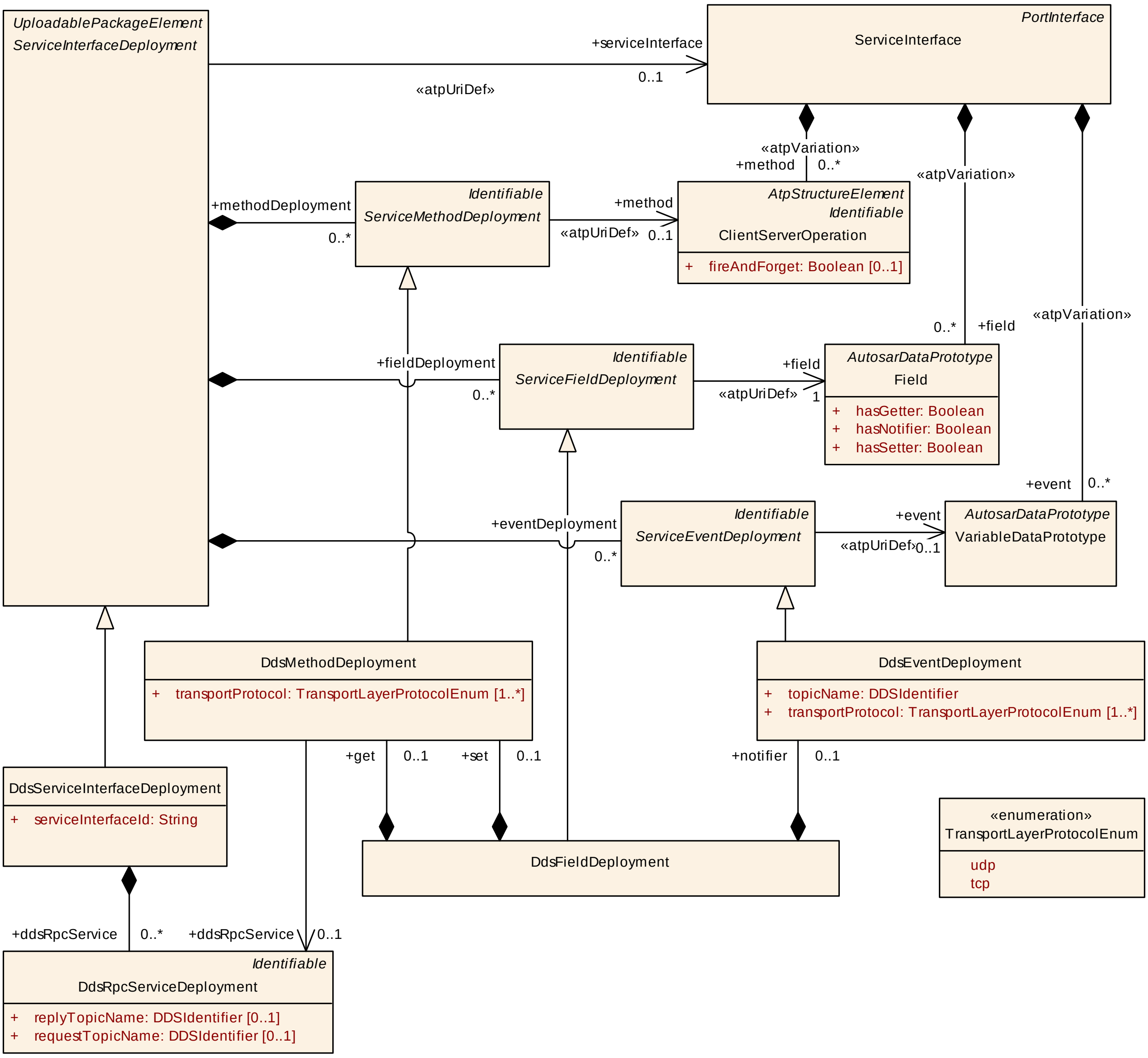


Figure 9.3: DDS deployment of ServiceInterface

[TPS_MANI_03525]{DRAFT} **DDS ServiceInterface binding** ⌈ The `DdsServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` to DDS and to assign a DDS Service identifier to the `ServiceInterface` with the `serviceInterfaceId` attribute. ⌋*(RS_MANI_00038)*

Class	**DdsServiceInterfaceDeployment**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment
Note	DDS configuration settings for a ServiceInterface. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInterfaceDeployments

▽

△

Class	DdsServiceInterfaceDeployment			
Base	*ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, ServiceInterfaceDeployment, UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
ddsRpcService	DdsRpcServiceDeployment	*	aggr	This aggregation represents the settings of DDS-RPC services associated with a Service Interface to handle methods and field getters and setters when using DDS as the underlying network binding. **Tags:** atp.Status=draft
serviceInterfaceId	String	1	attr	Unique Identifier that identifies the ServiceInterface in DDS. This Identifier is encoded in the USER_DATA QoS of the DomainParticipant associated with the Service Instance and its value is propagated by DDS Discovery messages. **Tags:** atp.Status=draft

Table 9.12: DdsServiceInterfaceDeployment

[TPS_MANI_03556]{DRAFT} **DDS-RPC Service Binding** ⌈ The `DdsRpcServiceDeployment` meta-class provides the ability to configure the name of the DDS Request and Reply Topics associated with a DDS-RPC Service with the `requestTopicName` and `replyTopicName` attributes, respectively. DDS-RPC Services are the mechanisms specified in the OMG RPC over DDS specification (DDS-RPC [24]) to handle method calls with DDS. The `requestTopicName` and `replyTopicName` attributes are optional, if unspecified they shall be configured as specified in the DDS-RPC specification. ⌋*(RS_MANI_00038)*

Class	DdsRpcServiceDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	Configuration settings for a DDS-RPC service capable of providing access to the methods and field getters/setters of a service interface. **Tags:** atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
replyTopicName	DDSIdentifier	0..1	attr	Name of the DDS Reply Topic associated with the Method. **Tags:** atp.Status=draft
requestTopicName	DDSIdentifier	0..1	attr	Name of the DDS Request Topic associated with the Method. **Tags:** atp.Status=draft

Table 9.13: DdsRpcServiceDeployment

[TPS_MANI_03526]{DRAFT} **DDS `VariableDataPrototype` binding** ⌈ The `DdsEventDeployment` meta-class provides the ability to bind a `VariableDataPrototype` to DDS and to assign a DDS Topic to the `event` with the `topicName` attribute. Moreover, the meta-class provides the ability to configure the `transportProtocols` over which the `VariableDataPrototype` may be accessed. ⌋*(RS_MANI_00038)*

Class	DdsEventDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for an Event. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceEventDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
topicName	DDSIdentifier	1	attr	Name of the DDS Topic associated with the Event. **Tags:** atp.Status=draft
transport Protocol	TransportLayerProtocol Enum	1..*	attr	This attribute defines over which Transport Layer Protocol(s) this event is intended to be sent. **Tags:** atp.Status=draft

Table 9.14: DdsEventDeployment

[TPS_MANI_03557]{DRAFT} **DDS ClientServerOperation Binding** ⌈ The DdsMethodDeployment meta-class provides the ability to bind a ClientServer-Operation to DDS and to assign it to a DDS-RPC Service. Moreover, the meta-class provides the ability to configure the transportProtocols over which the ClientServerOperation may be accessed. ⌋*(RS_MANI_00038)*

Class	DdsMethodDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for a Method. **Tags:** atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceMethodDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
ddsRpcService	DdsRpcService Deployment	0..1	ref	Configuration of the DDS-RPC service providing access to the method when using DDS as the underlying network binding. **Tags:** atp.Status=draft
transport Protocol	TransportLayerProtocol Enum	1..*	attr	This attribute defines over which Transport Layer Protocol(s) this method is intended to be sent. **Tags:** atp.Status=draft

Table 9.15: DdsMethodDeployment

[TPS_MANI_03558]{DRAFT} **DDS Field Binding** ⌈ The DdsFieldDeployment meta-class provides the ability to bind a Field to DDS, to assign its get and set ClientServerOperations to a DDS-RPC Service, and to bind its event notifier to a DDS Topic. ⌋*(RS_MANI_00038)*

Class	DdsFieldDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for a Field. **Tags:** atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceFieldDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
get	DdsMethodDeployment	0..1	aggr	This aggregation represents the setting of the get method. **Tags:** atp.Status=draft
notifier	DdsEventDeployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:** atp.Status=draft
set	DdsMethodDeployment	0..1	aggr	This aggregation represents the settings of the set method. **Tags:** atp.Status=draft

Table 9.16: DdsFieldDeployment

9.1.3 User Defined Service Interface

This chapter describes a user defined deployment of a `ServiceInterface` to a middleware technology that is not standardized by AUTOSAR. Such `UserDefinedServiceInterfaceDeployment` can for example also be used to describe a machine local IPC communication.

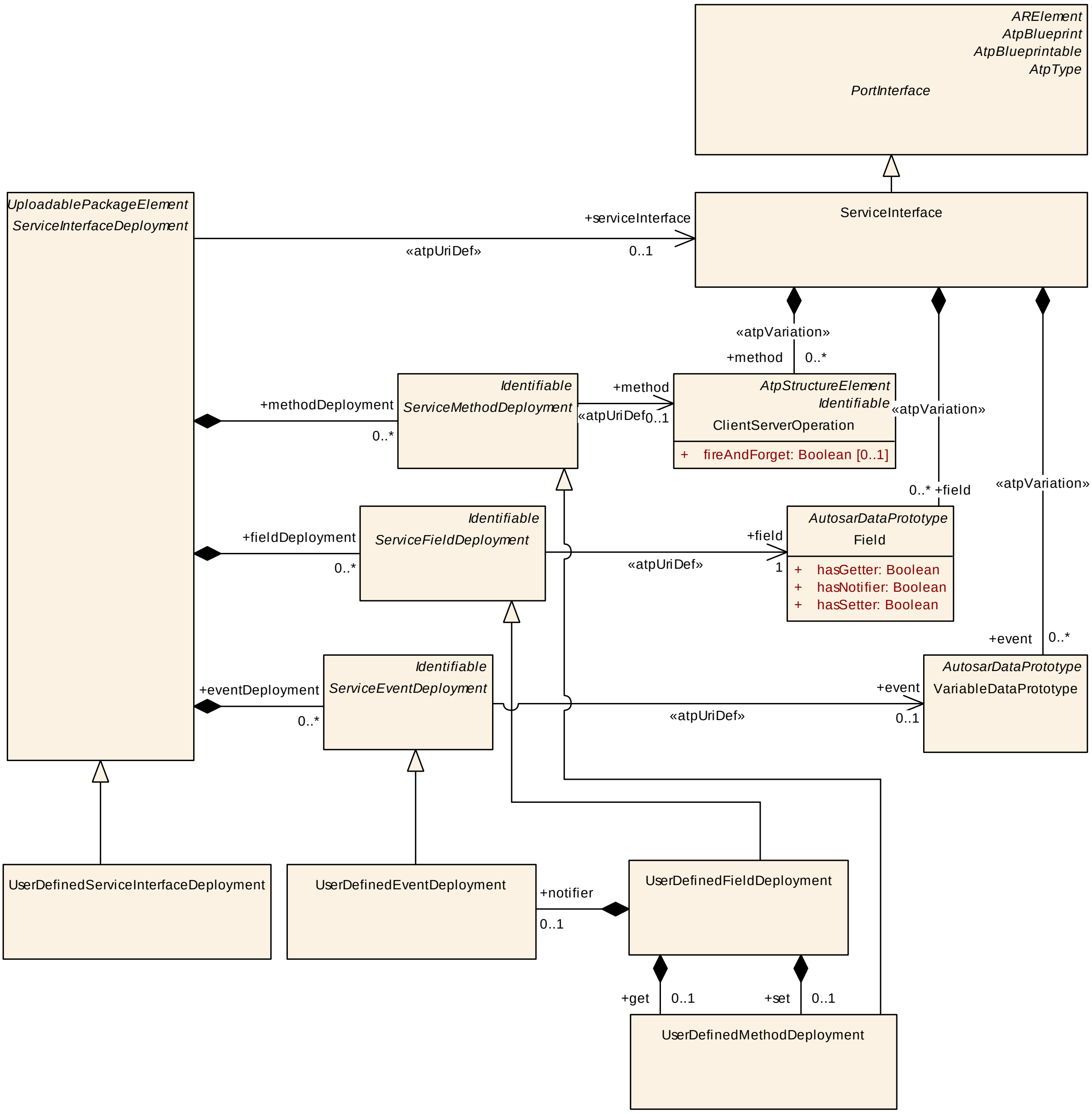


Figure 9.4: User defined deployment of ServiceInterface

[TPS_MANI_03045]{DRAFT} **UserDefined ServiceInterface binding** ⌈ The UserDefinedServiceInterfaceDeployment meta-class provides the ability to bind a ServiceInterface that is referenced in the role serviceInterface to a middleware technology that is not standardized by AUTOSAR. ⌋*(RS_MANI_00014)*

Please note that UserDefinedServiceInterfaceDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedServiceInterfaceDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a ServiceInterface. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInterfaceDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceDeployment*, *UploadablePackageElement*			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 9.17: UserDefinedServiceInterfaceDeployment

[TPS_MANI_01165]{DRAFT} **Standardized value of UserDefinedServiceInterfaceDeployment.category** ⌈ The AUTOSAR Standard reserves the following value for attribute UserDefinedServiceInterfaceDeployment.category:

- SERVICE_INTERFACE_DEPLOYMENT_IPC

It is possible to use a custom, non-standardized value for the attribute UserDefinedServiceInterfaceDeployment.category but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values. ⌋*(RS_MANI_00014)*

[constr_1570]{DRAFT} **Restriction for UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC** ⌈ An AdaptivePlatformServiceInstance that references a UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC shall not be referenced by a UserDefinedServiceInstanceToMachineMapping in the role serviceInstance. ⌋*()*

Rationale for [constr_1570]: for a local IPC binding it is not necessary to define the relation of the AdaptivePlatformServiceInstance to a CommunicationConnector.

[TPS_MANI_03046]{DRAFT} **User defined VariableDataPrototype binding** ⌈ The UserDefinedEventDeployment meta-class provides the ability to bind a VariableDataPrototype that is referenced in the role event to a middleware technology that is not standardized by AUTOSAR. ⌋*(RS_MANI_00014)*

Please note that UserDefinedEventDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedEventDeployment
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment
Note	UserDefined configuration settings for an Event. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft

▽

△

Class	UserDefinedEventDeployment			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceEventDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 9.18: UserDefinedEventDeployment

[TPS_MANI_03047]{DRAFT} **User defined ClientServerOperation binding** ⌈ The UserDefinedMethodDeployment meta-class provides the ability to bind a ClientServerOperation that is referenced in the role method to a middleware technology that is not standardized by AUTOSAR. ⌋*(RS_MANI_00014)*

Please note that UserDefinedMethodDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedMethodDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a Method. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceMethodDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 9.19: UserDefinedMethodDeployment

[TPS_MANI_03048]{DRAFT} **User defined Field binding** ⌈ The UserDefinedFieldDeployment meta-class provides the ability to bind a Field that is referenced in the role field to a middleware technology that is not standardized by AUTOSAR. ⌋*(RS_MANI_00014)*

Please note that UserDefinedFieldDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedFieldDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a Field. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceFieldDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
get	UserDefinedMethodDeployment	0..1	aggr	This aggregation represents the settings of the get method **Tags:** atp.Status=draft

▽

△

Class	UserDefinedFieldDeployment			
notifier	UserDefinedEvent Deployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:** atp.Status=draft
set	UserDefinedMethod Deployment	0..1	aggr	This aggregation represents the settings of the set method **Tags:** atp.Status=draft

Table 9.20: UserDefinedFieldDeployment

[constr_3417]{DRAFT} **UserDefinedEventDeployments aggregated by a UserDefinedFieldDeployment** ⌈ A UserDefinedEventDeployment that is aggregated by a UserDefinedFieldDeployment in the role notifier shall not reference a VariableDataPrototype in the role event. ⌋*()*

[constr_3418]{DRAFT} **UserDefinedMethodDeployments aggregated by a UserDefinedFieldDeployment** ⌈ A UserDefinedMethodDeployment that is aggregated by a UserDefinedFieldDeployment in the role get or set shall not reference a ClientServerOperation in the role method. ⌋*()*

9.2 Service Instance Deployment

An AdaptivePlatformServiceInstance makes the functionality of a ServiceInterface available on the *AUTOSAR adaptive platform*. Several AdaptivePlatformServiceInstances may be set up for the same ServiceInterface. They deliver the same functionality, but for different purposes and/or to different users.

The ProvidedApServiceInstance represents a provider that offers the functionality of a ServiceInterface with particular properties. Clients that are represented by the RequiredApServiceInstance observe offers and choose a provider with respect to service properties.

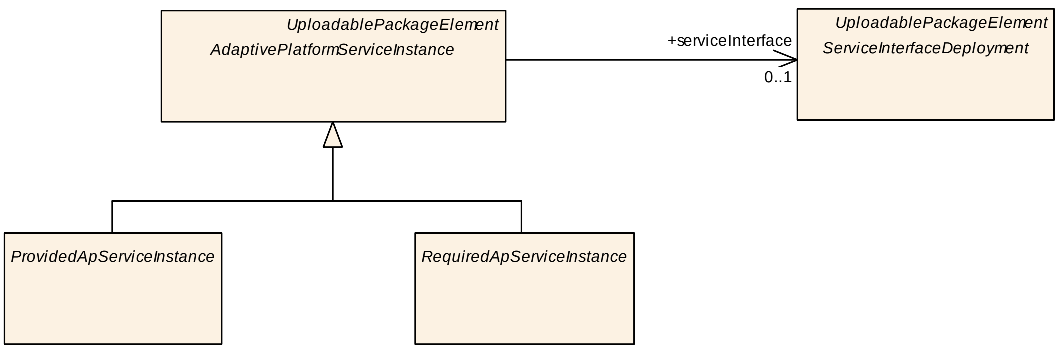


Figure 9.5: Modeling of the AdaptivePlatformServiceInstance

Note that the abstract meta-class AdaptivePlatformServiceInstance is derived from ARElement. This means that all meta-classes derived from AdaptivePlatformServiceInstance can be declared on the M1 level as part of an ARPackage and thus can be used in several different Manifest descriptions.

Class	***AdaptivePlatformServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a service instance in an abstract way. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	*ProvidedApServiceInstance*, *RequiredApServiceInstance*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
e2eEvent ProtectionProps	End2EndEvent ProtectionProps	*	aggr	This aggregation allows to protect an event or a field notifier that is defined inside of the ServiceInterface that is referenced by the ServiceInstance in the role service Interface. **Tags:** atp.Status=draft
secureCom Config	ServiceInterface ElementSecureCom Config	*	aggr	Configuration settings to secure the communication of ServiceInterface elements. **Tags:** atp.Status=draft
serviceInterface	ServiceInterface Deployment	0..1	ref	Reference to a ServiceInterfaceDeployment that identifies the ServiceInterface that is represented by the Service Instance. **Tags:** atp.Status=draft

Table 9.21: AdaptivePlatformServiceInstance

Class	***RequiredApServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in an abstract way. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Subclasses	DdsRequiredServiceInstance, RequiredSomeipServiceInstance, RequiredUserDefinedServiceInstance			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 9.22: RequiredApServiceInstance

Class	***ProvidedApServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in an abstract way. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Subclasses	DdsProvidedServiceInstance, ProvidedSomeipServiceInstance, ProvidedUserDefinedServiceInstance			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	***ProvidedApServiceInstance*** (abstract)			
–	–	–	–	–

Table 9.23: ProvidedApServiceInstance

There are two alternative ways to relate an AdaptivePlatformServiceInstance with a MachineDesign as described in [TPS_MANI_03000] and [TPS_MANI_03001]. Figure Figure 9.6 shows both approaches in an example.

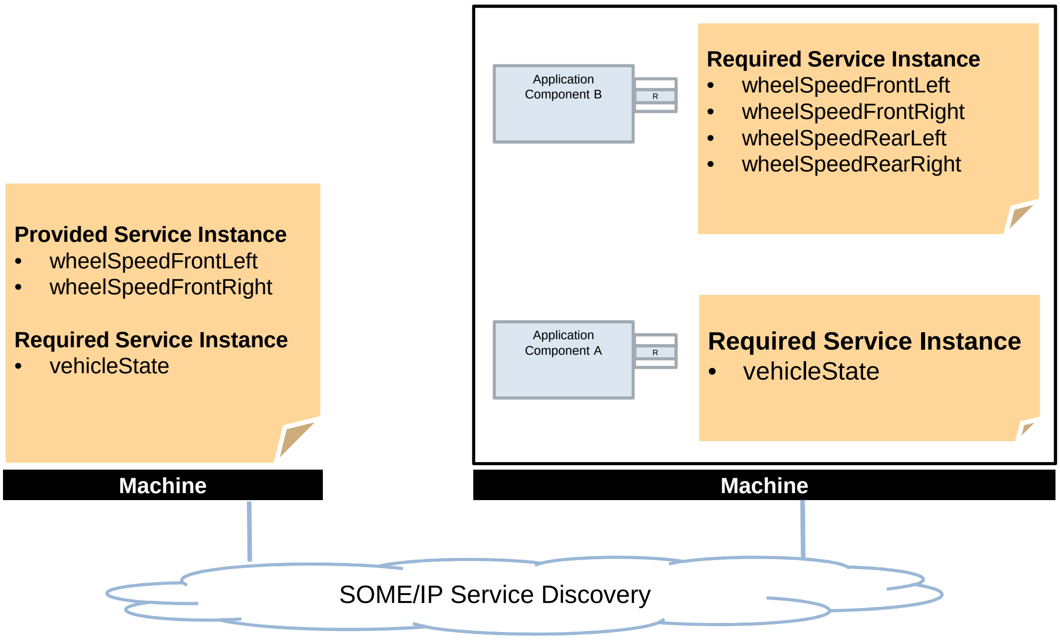


Figure 9.6: Different approaches for ServiceInstanceMapping

[TPS_MANI_03001]{DRAFT} **Mapping of AdaptivePlatformServiceInstance to a MachineDesign** ⌈ ServiceInstanceToMachineMapping is used to assign one or several AdaptivePlatformServiceInstances to (via a CommunicationConnector) a MachineDesign. This allows to define a "black box" machine view without any assumption on the application software but with all necessary information to configure the communication (e.g. SOME/IP). ⌋*(RS_MANI_00009)*

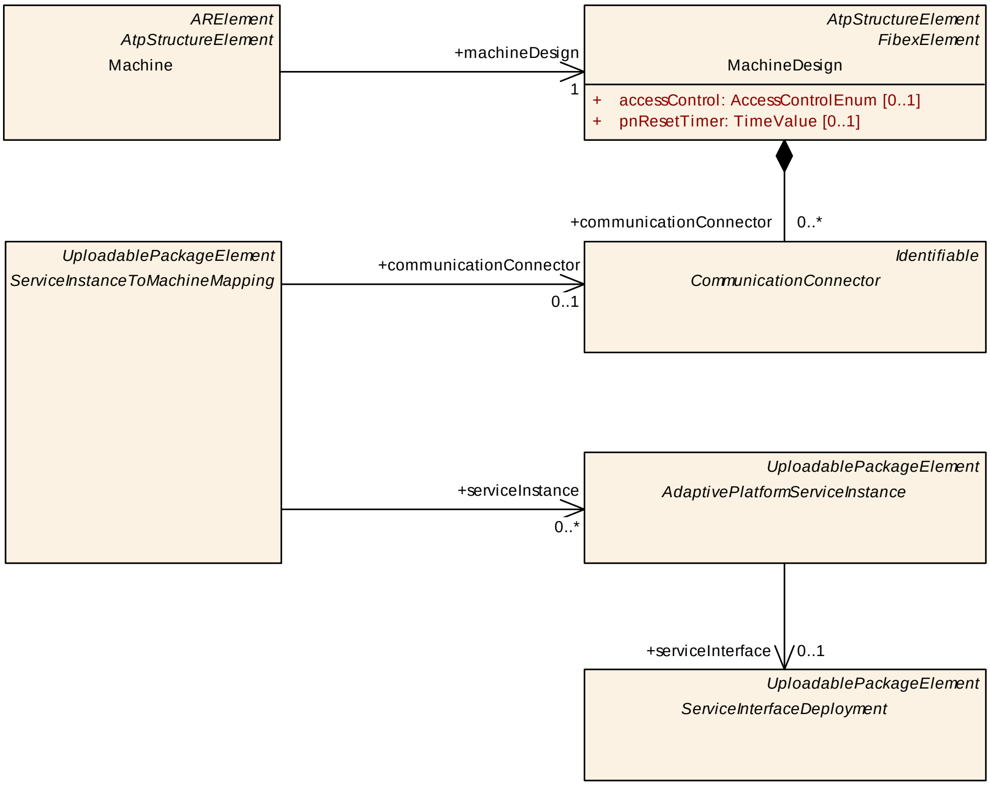


Figure 9.7: ServiceInstanceToMachineMapping

Class	***ServiceInstanceToMachineMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class represents the ability to map one or several AdaptivePlatformServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	DdsServiceInstanceToMachineMapping, SomeipServiceInstanceToMachineMapping, UserDefined ServiceInstanceToMachineMapping			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
communication Connector	Communication Connector	0..1	ref	Reference to the Machine to which the ServiceInstance is mapped. **Tags:** atp.Status=draft
secOcCom PropsFor Multicast	SecOcSecureCom Props	*	ref	Reference to communication security configuration settings that are valid for the udp multicast endpoint (Port + Multicast IP Address) defined by the ServiceInstanceTo MachineMapping. **Tags:** atp.Status=draft
secureCom PropsForTcp	SecureComProps	*	ref	Reference to communication security configuration settings that are valid for the tcp unicast endpoint (Tcp Port + Unicast IP Address) defined by the Service InstanceToMachineMapping. **Tags:** atp.Status=draft

▽

△

Class	ServiceInstanceToMachineMapping (abstract)			
secureCom PropsForUdp	SecureComProps	*	ref	Reference to communication security configuration settings that are valid for the udp unicast endpoint (Udp Port + Unicast IP Address) defined by the Service InstanceToMachineMapping. **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	Reference to a ServiceInstance that is mapped to the Machine. **Tags:** atp.Status=draft

Table 9.24: ServiceInstanceToMachineMapping

[constr_3297]{DRAFT} **SomeipServiceInstanceToMachineMapping only supports a single Address Family** ⌈A SomeipServiceInstanceToMachineMapping shall only support a single Address Family, i.e. either IPv4 or IPv6. The address family shall be consistent with the Ipv4Configuration/Ipv6Configuration of the NetworkEndpoint referenced by the EthernetCommunicationConnector that is referenced by the SomeipServiceInstanceToMachineMapping in the role communicationConnector.⌋*()*

[constr_3487]{DRAFT} **TCP endpoint can only serve provided or required service instances exclusively** ⌈ServiceInstanceToMachineMapping is not allowed to refer a ProvidedApServiceInstance and at the same time a RequiredApServiceInstance in the role serviceInstance if

- the ServiceInterfaceDeployment that is referenced by the ProvidedApServiceInstance in the role serviceInterface and
- the ServiceInterfaceDeployment that is referenced by the RequiredApServiceInstance in the role serviceInterface

both contain defined tcp content that is described by the transportProtocol attribute in the deployment elements of SOME/IP or DDS.

In other words a TCP endpoint can only serve provided or required service instances exclusively.⌋*()*

The reason for [constr_3487] is that the POSIX Socket API does not support the binding of several TCP sockets onto the same tuple <local IP address, local port>. But this would be necessary if a service is provided and consumed over the same TCP Endpoint.

[TPS_MANI_03000]{DRAFT} **Mapping of AdaptivePlatformServiceInstance to PortPrototypes** ⌈ServiceInstanceToPortPrototypeMapping is used to assign an AdaptivePlatformServiceInstance to a PortPrototype of a SwComponentType. This allows to define how specific PortPrototypes of a Software Component are represented in the middleware in terms of the service configuration.⌋*(RS_MANI_00011)*

In other words, the “outside” appearance of a PortPrototype from the middleware point of view is the AdaptivePlatformServiceInstance, or the concrete subclasses RequiredApServiceInstance and ProvidedApServiceInstance.

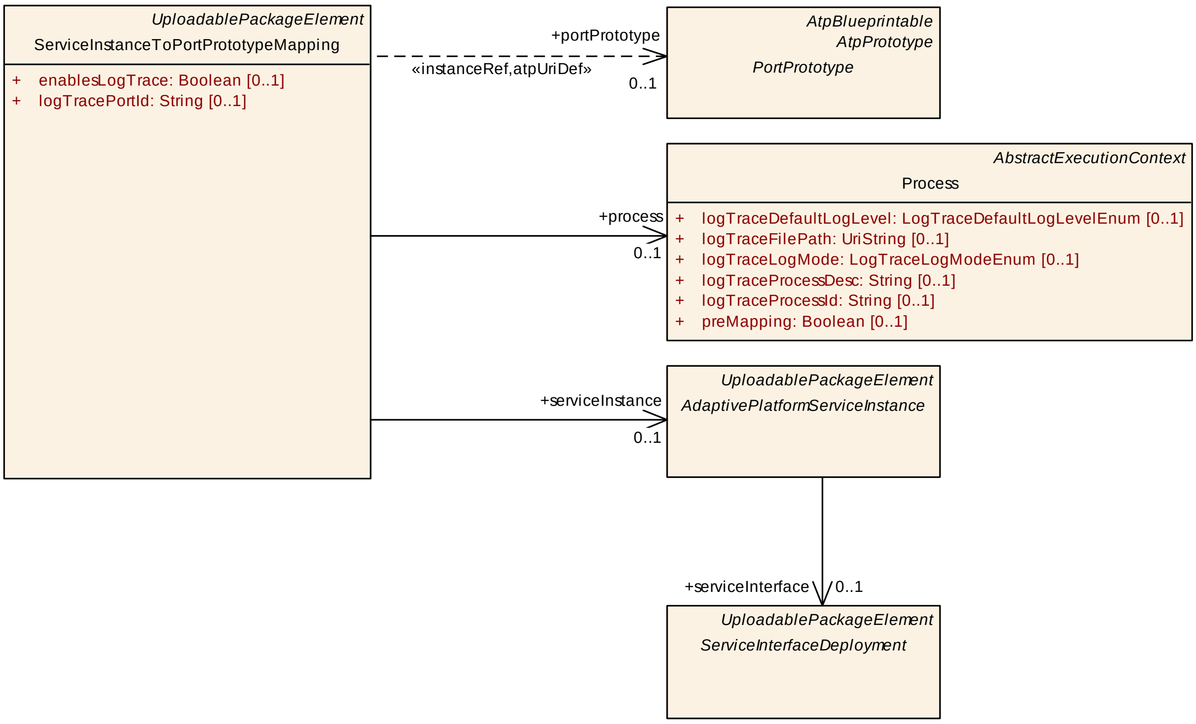


Figure 9.8: ServiceInstanceToPortPrototypeMapping

Class	**ServiceInstanceToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class represents the ability to assign a transport layer dependent ServiceInstance to a Port Prototype. With this mapping it is possible to define how specific PortPrototypes are represented in the middleware in terms of service configuration. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=ServiceInstanceToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
enablesLog Trace	Boolean	0..1	attr	This attribute enables/disables Log&Trace for the communication on the referenced Port of the referenced process. True: Log&Trace is enabled. False: Log&Trace is disabled.
logTracePortId	String	0..1	attr	This attribute identifies a Port of an Application executed in a process for tracing (ContextId).
portPrototype	PortPrototype	0..1	iref	Reference to a specific PortPrototypes that represents the ServiceInstance. **Tags:** atp.Status=draft

▽

△

Class	ServiceInstanceToPortPrototypeMapping			
process	Process	0..1	ref	Reference to the Process in which the Executable that contains the SoftwareComponent and the referenced Port Prototype is executed. **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	Reference to a ServiceInstance that is represented in the Software Component by the mapped group of Port Prototypes. **Tags:** atp.Status=draft

Table 9.25: ServiceInstanceToPortPrototypeMapping

Meta-classes `ProvidedApServiceInstance` and `RequiredApServiceInstance` are abstract and this allows for using specific derived classes that fit the underlying middleware (e.g. SOME/IP). The following sub-chapters will detail the supported specializations.

9.2.1 SOME/IP Service Instance Deployment

In the case of SOME/IP used as the middleware the derived meta-classes are `ProvidedSomeipServiceInstance` or `RequiredSomeipServiceInstance`. These meta-classes also carry attributes that apply for the service discovery on SOME/IP.

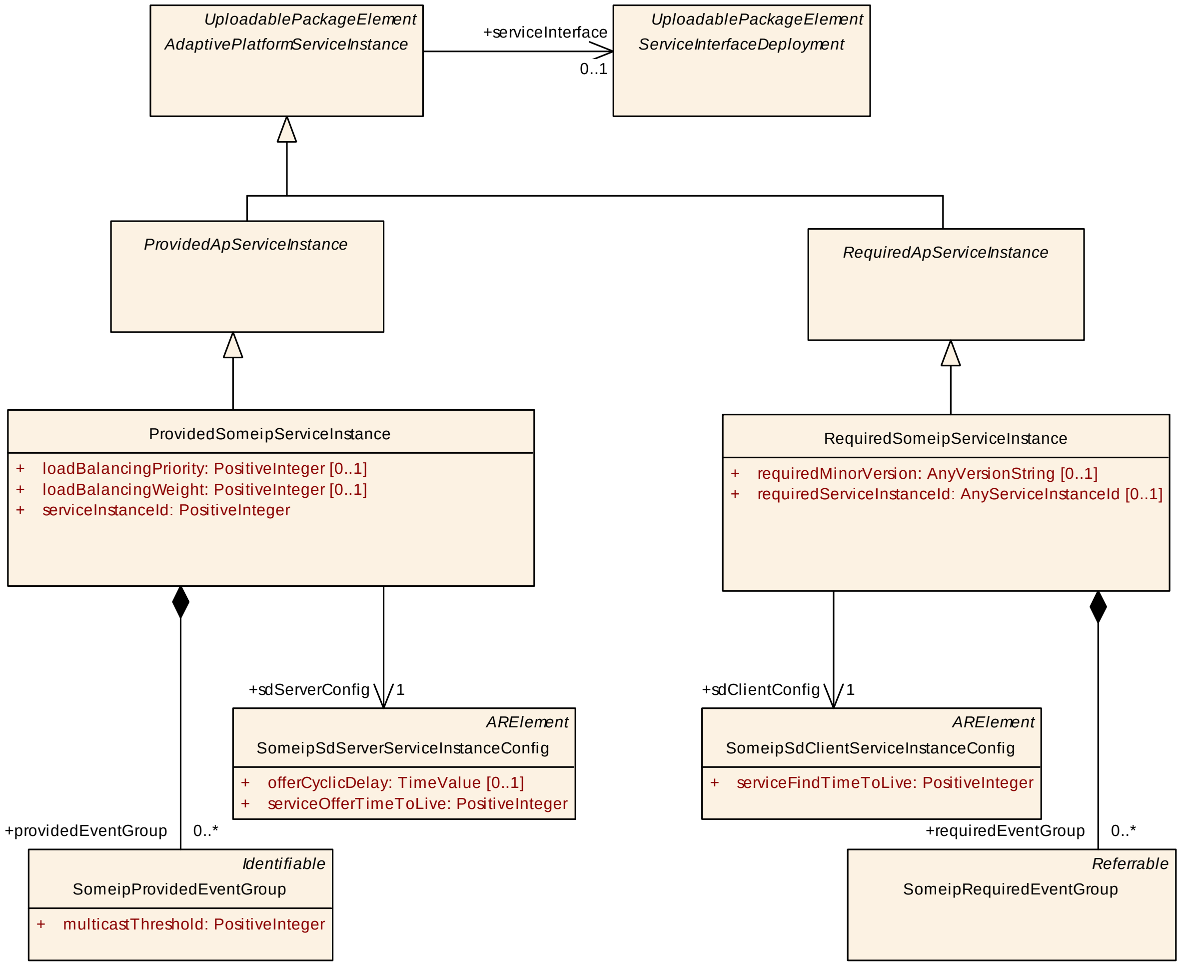


Figure 9.9: SOME/IP Service Instances

Primitive	**AnyServiceInstanceId**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This is a positive integer or the literal ANY which can be denoted in decimal, octal and hexadecimal. The value is between 0 and 4294967295. **Tags:** atp.Status=draft xml.xsd.customType=ANY-SERVICE-INSTANCE-ID xml.xsd.pattern=[1-9][0-9]*|0[xX][0-9a-fA-F]+|0[0-7]*|0[bB][0-1]+|ANY xml.xsd.type=string

Table 9.26: AnyServiceInstanceId

9.2.1.1 Provided Service Instance

The `ProvidedSomeipServiceInstance` defines the `serviceInstanceId` for the Service Instance of the `SomeipServiceInterfaceDeployment` that is referenced with the `serviceInterface` reference.

It means that the Server on which the ProvidedSomeipServiceInstance is deployed offers the Service Instance over SOME/IP with the serviceInstanceId and serviceInterfaceId.

Class	**ProvidedSomeipServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *Uploadable PackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
capability Record (ordered)	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Tags:** atp.Status=draft
eventProps	SomeipEventProps	*	aggr	Configuration settings for individual events that are provided by the ServiceInstance. **Tags:** atp.Status=draft
loadBalancing Priority	PositiveInteger	0..1	attr	This attribute is used to specify the priority in the load balancing option of SOME/IP that is added to the Offer Service. When a client searches for all service instances of a service, the client shall choose the service instance with highest priority if one is defined.
loadBalancing Weight	PositiveInteger	0..1	attr	This attribute is used to specify the weight in the load balancing option of SOME/IP that is added to the Offer Service. When a client searches for all service instances of a service, the client shall choose the service instance with highest priority if one is defined. If several service instances exist with the highest priority the service instance shall be chosen based on the weights of the service instances.
method ResponseProps	SomeipMethodProps	*	aggr	Configuration settings for individual methods that are provided by the ServiceInstance. **Tags:** atp.Status=draft
providedEvent Group	SomeipProvidedEvent Group	*	aggr	List of EventGroups that are provided by the Service Instance. **Tags:** atp.Status=draft
sdServerConfig	SomeipSdServer ServiceInstanceConfig	1	ref	Server specific configuration settings relevant for the SOME/IP service discovery. **Tags:** atp.Status=draft
serviceInstance Id	PositiveInteger	1	attr	Identification number that is used by SOME/IP service discovery to identify the instance of the service.

Table 9.27: ProvidedSomeipServiceInstance

[constr_3287]{DRAFT} **Mandatory information of a ProvidedSomeipServiceInstance** ⌈ The ProvidedSomeipServiceInstance shall always define the serviceInstanceId. ⌋*()*

In addition to the service identification properties a SOME/IP offer message contains so called endpoint options that define how the service instance is reachable by clients.

[TPS_MANI_03168]{DRAFT} **Configuration of the SOME/IP load balancing option** ⌈ The SOME/IP load balancing option is configurable per ProvidedSomeipServiceInstance with the two attributes loadBalancingPriority and loadBalancingWeight. ⌋*(RS_MANI_00024)*

The SOME/IP load balancing option is used to prioritize different ProvidedSomeipServiceInstances that point to the same SomeipServiceInterfaceDeployment, so that a client chooses the service instance based on these settings. This option is attached to SOME/IP Offer Service entries.

[constr_3415]{DRAFT} **Value range of loadBalancingPriority** ⌈ The value of loadBalancingPriority shall be in the range of 0..65535. ⌋*()*

Please note that according to SOME/IP a lower value means higher priority.

[constr_3416]{DRAFT} **Value range of loadBalancingWeight** ⌈ The value of loadBalancingWeight shall be in the range of 0..65535. ⌋*()*

Please note that according to SOME/IP a higher value means higher probability to be chosen.

9.2.1.1.1 IP Configuration

In SOME/IP the Offer service entry references IPv4 or IPv6 Endpoint options to indicate to the client where the server accepts the method calls and where the server sends the event messages.

Such an Endpoint contains the IP address of the sender. The IP address configuration is described in this chapter.

[TPS_MANI_03002]{DRAFT} **IP configuration for a ProvidedSomeipServiceInstance** ⌈ A ProvidedSomeipServiceInstance can be mapped to a CommunicationConnector of a MachineDesign with the SomeipServiceInstanceToMachineMapping.

With this mapping an assignment of the ProvidedSomeipServiceInstance to a unicast IP Address is established since the EthernetCommunicationConnector refers to a NetworkEndpoint in the role unicastNetworkEndpoint. ⌋ *(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03003]{DRAFT} **ProvidedSomeipServiceInstance Fanout** ⌈ It is allowed to map the same ProvidedSomeipServiceInstance to different CommunicationConnectors of a MachineDesign. In such a case, several SomeipServiceInstanceToMachineMappings shall be defined.

This allows for offering the same ProvidedSomeipServiceInstance on different VLANs or even on different CommunicationClusters. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3538]{DRAFT} **Only one ServiceInstanceToMachineMapping per technology and CommunicationConnector** ⌈ Each AdaptivePlatformServiceInstance shall only be referenced up to once by a specific ServiceInstanceToMachineMapping subclass in the role serviceInstance where the ServiceInstanceToMachineMapping refer to the same CommunicationConnector. ⌋*()*

In other words, it is not allowed to define for the same service instance two ServiceInstanceToMachineMapping of the same kind (e.g. SomeipServiceInstanceToMachineMapping) which refer to the same CommunicationConnector.

[TPS_MANI_03554]{DRAFT} **Several SomeipServiceInstanceToMachineMappings with equal settings** ⌈ If

- one SomeipServiceInstanceToMachineMapping refers to several service instances in the role serviceInstance
- several SomeipServiceInstanceToMachineMappings with equal settings refer to several service instances in the role serviceInstance
- the combination of the two above applies

then for all of the referenced service instances the same network connection (i.e. Ethernet socket) will be used. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03555]{DRAFT} **Mix of SomeipServiceInstanceToMachineMapping and signal-based communication** ⌈ SomeipServiceInstanceToMachineMapping defines service instance communication on a specific Ethernet socket and the same socket may also be used for signal-based communication at the same time. ⌋*(RS_MANI_00009, RS_MANI_00024)*

Please note that the signal-based communication is described in section 10.

Via the definition of respective ISignalTriggering, PduTriggering, and SocketConnection for signal-based communication, the same values for Ethernet address and port may be defined as used at the SomeipServiceInstanceToMachineMapping.

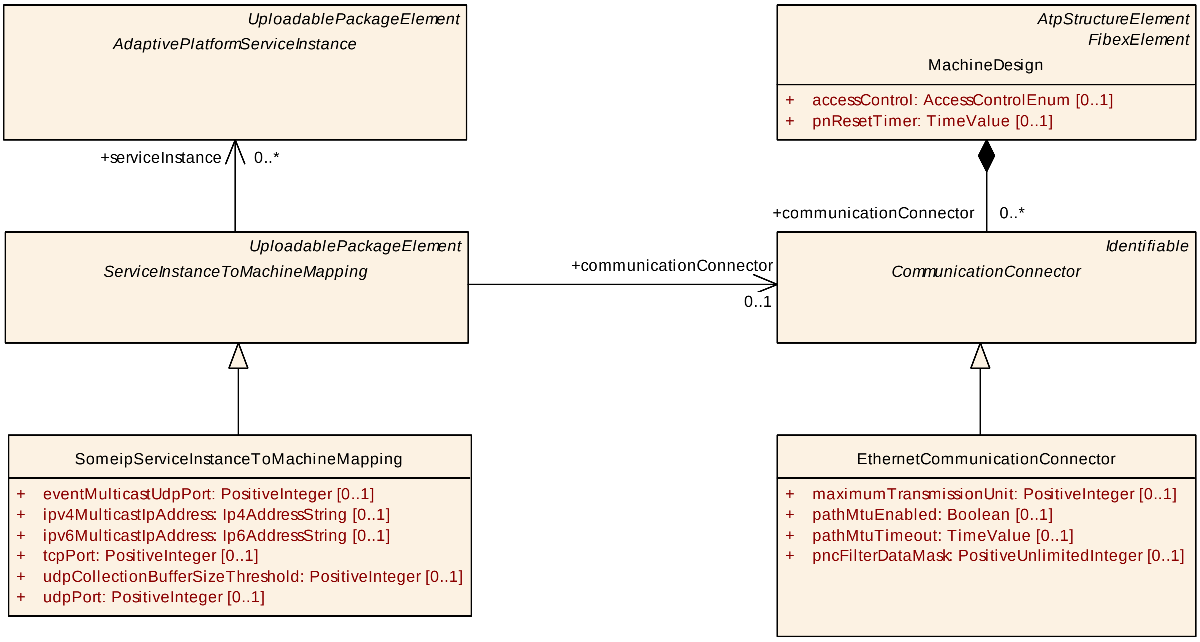


Figure 9.10: `SomeipServiceInstanceToMachineMapping` with TP and IP configuration

Class	«atpVariation» ***CommunicationCluster*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	The CommunicationCluster is the main element to describe the topological connection of communicating ECUs. A cluster describes the ensemble of ECUs, which are linked by a communication medium of arbitrary topology (bus, star, ring, ...). The nodes within the cluster share the same communication protocol, which may be event-triggered, time-triggered or a combination of both. A CommunicationCluster aggregates one or more physical channels. **Tags:** vh.latestBindingTime=postBuild			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	*AbstractCanCluster*, EthernetCluster, FlexrayCluster, LinCluster, UserDefinedCluster			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
baudrate	PositiveUnlimitedInteger	0..1	attr	Channels speed in bits/s.
physical Channel	PhysicalChannel	1..*	aggr	This relationship defines which channel element belongs to which cluster. A channel must be assigned to exactly one cluster, whereas a cluster may have one or more channels. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
protocolName	String	0..1	attr	The name of the protocol used.
protocolVersion	String	0..1	attr	The version of the protocol used.

Table 9.28: CommunicationCluster

Class	***CommunicationConnector*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	The connection between the referencing ECU and the referenced channel via the referenced controller. Connectors are used to describe the bus interfaces of the ECUs and to specify the sending/receiving behavior. Each CommunicationConnector has a reference to exactly one communicationController. Note: Several CommunicationConnectors can be assigned to one PhysicalChannel in the scope of one ECU Instance. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanCommunicationConnector*, EthernetCommunicationConnector, FlexrayCommunicationConnector, LinCommunicationConnector, UserDefinedCommunicationConnector			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 9.29: CommunicationConnector

Class	**EthernetCommunicationConnector**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet specific attributes to the CommunicationConnector. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
maximumTransmissionUnit	PositiveInteger	0..1	attr	This attribute specifies the maximum transmission unit in bytes.
networkEndpoint	NetworkEndpoint	*	ref	NetworkEndpoints
pathMtuEnabled	Boolean	0..1	attr	If enabled the IPv4/IPv6 processes incoming ICMP "Packet Too Big" messages and stores a MTU value for each destination address.
pathMtuTimeout	TimeValue	0..1	attr	If this value is >0 the IPv4/IPv6 will reset the MTU value stored for each destination after n seconds.
pncFilterDataMask	PositiveUnlimitedInteger	0..1	attr	Bit mask for Ethernet Payload used to configure the Ethernet Transceiver for partial network wakeup.
unicastNetworkEndpoint	NetworkEndpoint	0..1	ref	Network Endpoint that defines the IPAddress of the machine. **Tags:** atp.Status=draft

Table 9.30: EthernetCommunicationConnector

[constr_3288]{DRAFT} **IP configuration restriction for `unicastNetworkEndpoints`** ⌈ A `NetworkEndpoint` that is referenced by a `EthernetCommunicationConnector` in the role `unicastNetworkEndpoint` shall have either

- `Ipv4Configuration` or
- `Ipv6Configuration`

as `networkEndpointAddress` that is defined in the unicast IP range according to the rules defined in [TPS_MANI_03005] and [TPS_MANI_03006]. ⌋*()*

In SOME/IP, a server that offers a `ProvidedSomeipServiceInstance` is able to send events and notification events to an IP-Multicast address.

To indicate to the client to which Multicast IP address the event messages are send the Subscribe Eventgroup Acknowledgement entry contains a reference an IPv4 Multicast Option and/or and IPv6 Multicast Option.

[TPS_MANI_03004]{DRAFT} **IPv4 Multicast event destination address** ⌈ Meta-class SomeipServiceInstanceToMachineMapping defines the multicast IPv4 address to which the events and notification events are send with the attribute ipv4MulticastIpAddress. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03061]{DRAFT} **IPv6 Multicast event destination address** ⌈ Meta-class SomeipServiceInstanceToMachineMapping defines the multicast IPv6 address to which the events and notification events are sent with the attribute ipv6MulticastIpAddress. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03005]{DRAFT} **IPv4 Multicast address range** ⌈ The IPv4 addresses reserved for multicast communication are in the range 224.0.0.0 through 239.255.255.255. Addresses between 0.0.0.0 and 223.255.255.255 are reserved for unicast communication. ⌋*()*

[TPS_MANI_03006]{DRAFT} **IPv6 Multicast address range** ⌈ IPv6 multicast addresses are distinguished from unicast addresses by the value of the high-order octet of the addresses: a value of 0xFF (binary 11111111) identifies an address as an address reserved for multicast communication; any other value identifies an address as a unicast address. ⌋*()*

Class	***NetworkEndpointAddress*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	To build a valid network endpoint address there has to be either one MAC multicast group reference or an ipv4 configuration or an ipv6 configuration. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*			
Subclasses	Ipv4Configuration, Ipv6Configuration, MacMulticastConfiguration			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 9.31: NetworkEndpointAddress

9.2.1.1.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP Offer message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the sender.

With the SomeipServiceInstanceToMachineMapping the Transport Layer configuration attributes are assigned to the ProvidedSomeipServiceInstance.

The same element contains the Transport Layer configuration attributes for the IPv4/IPv6 Multicast Option that may be used in the SOME/IP SubscribeEventGroupAck message.

[TPS_MANI_03007]{DRAFT} **Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance** ⌈ The attribute SomeipServiceInstanceToMachineMapping.udpPort defines the Transport Protocol for a UDP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an OfferService entry. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03008]{DRAFT} **Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance** ⌈ The attribute SomeipServiceInstanceToMachineMapping.tcpPort defines the Transport Protocol for a TCP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an OfferService entry. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03009]{DRAFT} **Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance** ⌈ It is allowed to set tcpPort and udpPort in the same SomeipServiceInstanceToMachineMapping.

Such a setting shall be used to indicate that one UDP endpoint and one TCP endpoint are referenced in the OfferService entry. It means that the Server provides the ProvidedSomeipServiceInstance over both Transport Protocols. ⌋*(RS_MANI_00009, RS_MANI_00024)*

If a Tcp and Udp Transport Protocol Configuration is defined for a ProvidedSomeipServiceInstance as described in [TPS_MANI_03009] then the SOME/IP ServiceInterfaceDeployment settings decide which content of the ProvidedSomeipServiceInstance is transported over udp and which content is transported over tcp.

This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

[TPS_MANI_03010]{DRAFT} **Udp Transport Protocol Configuration in case of IP-Multicast** ⌈ The SomeipServiceInstanceToMachineMapping.eventMulticastUdpPort defines the Transport Protocol Port Number for a UDP event communication in case IP-Multicast is used.

This setting is used in an IPv4 or IPv6 Multicast Option that is referenced by a SubscribeEventGroupAck Service entry. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3290]{DRAFT} **Transport Protocol attributes defined for a ProvidedSomeipServiceInstance** ⌈ Each SomeipServiceInstanceToMachineMapping that is defined for a ProvidedSomeipServiceInstance shall define either

- a udpPort or
- a tcpPort or
- a udpPort and a tcpPort.

⌋*()*

[TPS_MANI_03157]{DRAFT} **Enabling of data collection for upd data transmission** ⌈ The setting of the attribute SomeipServiceInstanceToMachineMapping.udpCollectionBufferSizeThreshold to a value enables the data collection for data transmission over the udpPort and unicastNetworkEndpoint defined on the EthernetCommunicationConnector that is referenced by the SomeipServiceInstanceToMachineMapping. In this case all event and method messages that are configured for data collection will be collected in the buffer until a transmission trigger arrives and the data transmission starts. ⌋*(RS_MANI_00024)*

For configuration of transmission triggers please see [TPS_MANI_03158] and [TPS_MANI_03159].

9.2.1.1.3 Service Discovery Server Configuration

The multicast messages of the SOME/IP Service Discovery come with the risk of overflowing Machines with too many messages. Therefore, the Service Discovery can be configured with a suitable message sending behavior.

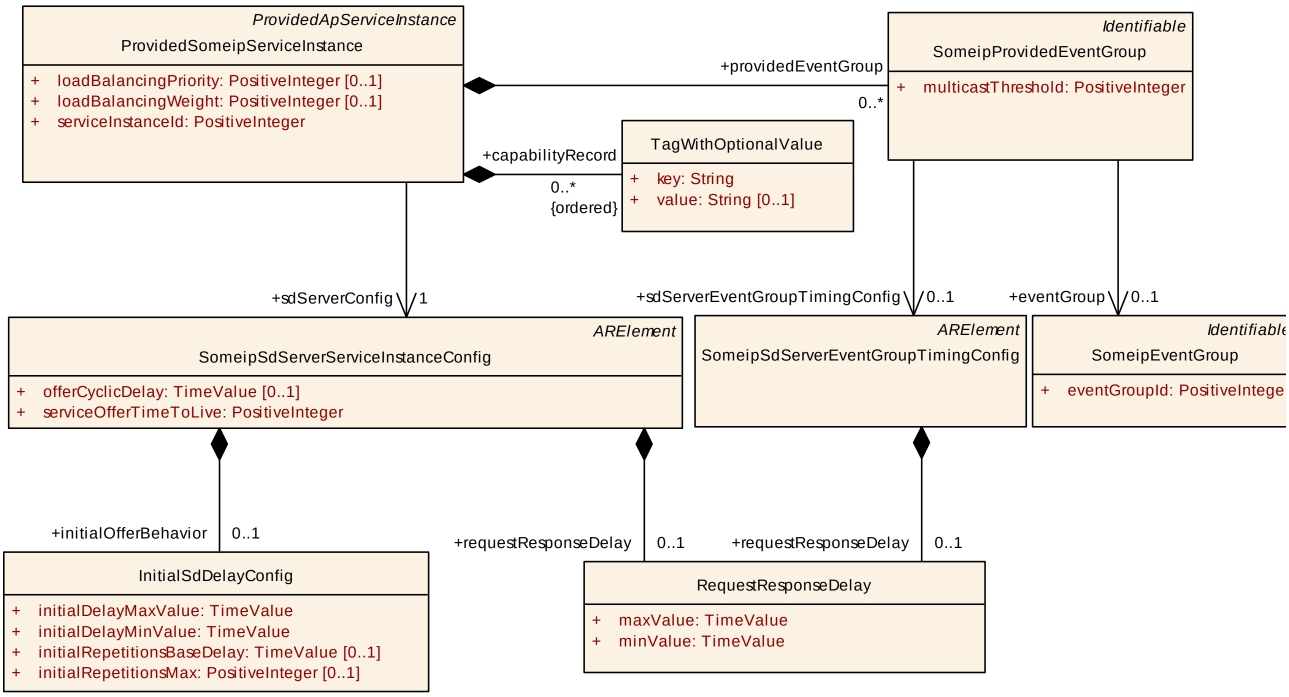


Figure 9.11: SOME/IP Service Discovery Server configuration settings

For every ProvidedSomeipServiceInstance on a Server different phases are existing:

- Down
- Available
  - Initial Wait Phase

– Repetition Phase

– Main Phase

[TPS_MANI_03011]{DRAFT} **Server Timing configuration for a ProvidedSomeipServiceInstance** ⌈ The Server Timing is configurable with SomeipSdServerServiceInstanceConfig that is referenced in the role sdServerConfig by the ProvidedSomeipServiceInstance for which the Timing is valid. ⌋ *(RS_MANI_00024)*

Class	SomeipSdServerServiceInstanceConfig			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Server specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
initialOfferBehavior	InitialSdDelayConfig	0..1	aggr	Controls offer behavior of the server. **Tags:** atp.Status=draft
offerCyclicDelay	TimeValue	0..1	attr	Optional attribute to define cyclic offers. Cyclic offer is active, if the delay is set (in seconds).
requestResponseDelay	RequestResponseDelay	0..1	aggr	Maximum/Minimum allowable response delay to entries received by multicast in seconds. The Service Discovery shall delay answers to entries that were transported in a multicast SOME/IP-SD message (e.g. FindService). **Tags:** atp.Status=draft
serviceOfferTimeToLive	PositiveInteger	1	attr	Defines the time in seconds the service offer is valid.

Table 9.32: SomeipSdServerServiceInstanceConfig

[TPS_MANI_03012]{DRAFT} **Initial Wait Phase configuration for a ProvidedSomeipServiceInstance** ⌈ The Initial Wait Phase for a ProvidedSomeipServiceInstance is configured with the initialOfferBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue.

When a calculated random timer based on these min and max values expires the first OfferService entry will be sent out. ⌋ *(RS_MANI_00024)*

When the calculated random timer expires the Repetition Phase will be entered.

[TPS_MANI_03013]{DRAFT} **Repetition Wait Phase configuration for a ProvidedSomeipServiceInstance** ⌈ The Repetition Wait Phase for a ProvidedSomeipServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay. ⌋ *(RS_MANI_00024)*

If the Repetition Phase is entered the Service Discovery waits for the initialRepetitionsBaseDelay and then sends an OfferService entry. If the amount of sent

OfferService entries reaches initialRepetitionsMax the Main Phase will be entered.

[TPS_MANI_03014]{DRAFT} **Main Phase configuration for a ProvidedSomeipServiceInstance** ⌈ The Main Phase for a ProvidedSomeipServiceInstance is configured with the offerCyclicDelay attribute of SomeipSdServerServiceInstanceConfig.

The OfferService entry will be sent cyclically with an interval that is defined by the value of attribute offerCyclicDelay. ⌋*(RS_MANI_00024)*

Class	**InitialSdDelayConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This element is used to configure the offer behavior of the server and the find behavior on the client. **Tags:** atp.ManifestKind=ServiceInstanceManifest			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
initialDelayMaxValue	TimeValue	1	attr	Max Value in seconds to delay randomly the first offer (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the transmission of a find message (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).
initialDelayMinValue	TimeValue	1	attr	Min Value in seconds to delay randomly the first offer (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the transmission of a find message (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).
initialRepetitionsBaseDelay	TimeValue	0..1	attr	The base delay for offer repetitions (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or find repetitions (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig). Successive find messages have an exponential back off delay.
initialRepetitionsMax	PositiveInteger	0..1	attr	Describes the maximum amount of offer repetitions (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the maximum amount of find repetitions (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).

Table 9.33: InitialSdDelayConfig

[TPS_MANI_03015]{DRAFT} **TTL for Offer Service Entries** ⌈ The lifetime of a ProvidedSomeipServiceInstance is configurable with the serviceOfferTimeToLive attribute of SomeipSdServerServiceInstanceConfig.

If the time that is configured by serviceOfferTimeToLive expires the ProvidedSomeipServiceInstance is no longer offered. ⌋*(RS_MANI_00024)*

[TPS_MANI_03016]{DRAFT} **Servers RequestResponseDelay for received FindService entries** ⌈ The Server will delay the OfferService answer to a received multicast FindService entry by the configured SomeipSdServerServiceInstanceConfig.requestResponseDelay.

The actual delay will be randomly chosen between the maxValue and minValue. ⌋*(RS_MANI_00024)*

Class	RequestResponseDelay			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Time to wait before answering the query. **Tags:** atp.ManifestKind=ServiceInstanceManifest			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
maxValue	TimeValue	1	attr	Maximum allowable response delay to entries received by multicast in seconds.
minValue	TimeValue	1	attr	Minimum allowable response delay to entries received by multicast in seconds.

Table 9.34: RequestResponseDelay

Figure 9.12 shows an example of the different SOME/IP phases on the Server side.

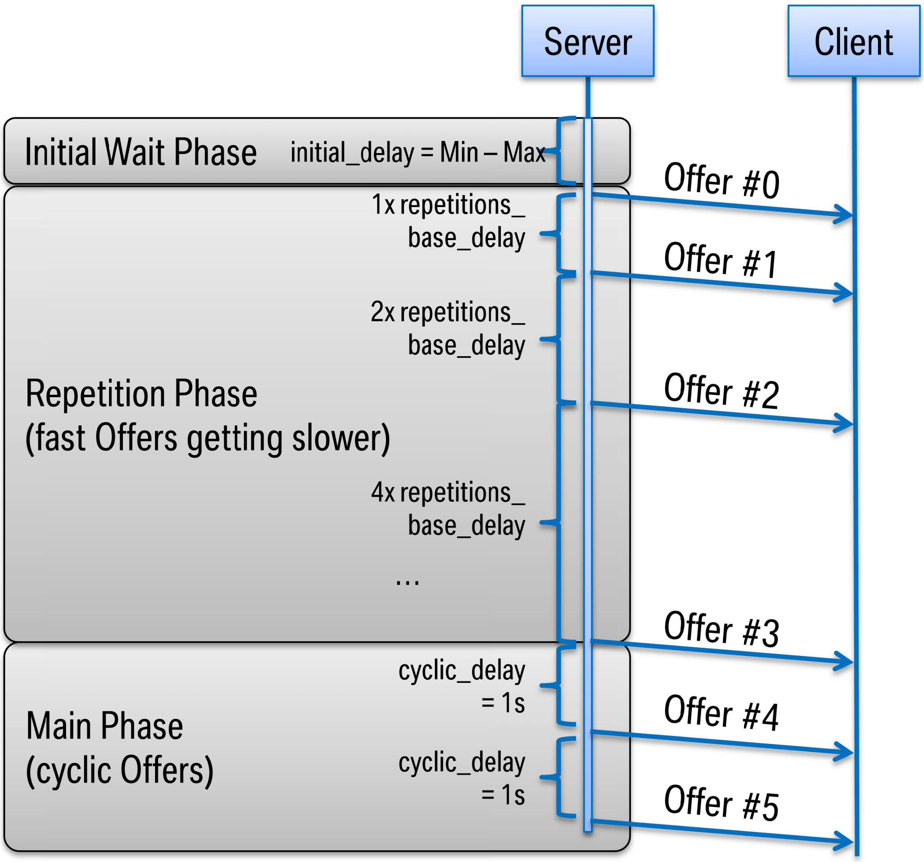


Figure 9.12: SOME/IP Server Timing example

SOME/IP allows for the specification of additional information about the ProvidedSomeipServiceInstance with the Capability Record that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.

[TPS_MANI_03017]{DRAFT} **Server Capability Records** ⌈ A Capability Record (key/value pair) on the Server side is configurable with the capabilityRecord and the two attributes key and value. ⌋*(RS_MANI_00024)*

Class	TagWithOptionalValue			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::TagWithOptionalValue			
Note	A tagged value is a combination of a tag (key) and a value that gives supplementary information that is attached to a model element. Please note that keys without a value are allowed. **Tags:** atp.ManifestKind=ServiceInstanceManifest			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
key	String	1	attr	Defines a key.
value	String	0..1	attr	Defines the corresponding value.

Table 9.35: TagWithOptionalValue

9.2.1.1.4 Provided Event Group

The ProvidedSomeipServiceInstance aggregates a SomeipProvidedEventGroup in the role providedEventGroup that allows to define service instance specific configuration settings for a SomeipEventGroup.

Class	SomeipProvidedEventGroup			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the provided side for each EventGroup separately. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related Event Group settings are valid. **Tags:** atp.Status=draft
multicast Threshold	PositiveInteger	1	attr	Specifies the number of subscribed clients that trigger the server to change the transmission of events to multicast. Example: If configured to 0 only unicast will be used. If configured to 1 the first client will be already served by multicast. If configured to 2 the first client will be server with unicast and as soon as the 2nd client arrives both will be served by multicast. This does not influence the handling of initial events, which are served using unicast only.
sdServerEvent GroupTiming Config	SomeipSdServerEvent GroupTimingConfig	0..1	ref	Server Timing configuration settings that are EventGroup specific. **Tags:** atp.Status=draft

Table 9.36: SomeipProvidedEventGroup

[TPS_MANI_03018]{DRAFT} **Usage of SomeipProvidedEventGroup.multicastThreshold** ⌈ The switching between IP-Unicast and IP-Multicast is guided by the server with the SomeipProvidedEventGroup.multicastThreshold attribute and by the number of subscribed clients to the SomeipProvidedEventGroup.

The Server will change the transmission of events to Multicast if the multicastThreshold of the corresponding SomeipProvidedEventGroup is reached by the number of subscribed clients. If the number of subscribed clients is smaller then the configured multicastThreshold, the transmission of events takes place via unicast communication. ⌋*(RS_MANI_00024)*

The following example shows the effect of the multicastThreshold in relation to the number of subscribed clients to the transmission of the SOME/IP event to the unicast or multicast destination address:

- If multicastThreshold is configured to 0 only the unicast IP address and the port will be used as destination address.
- If multicastThreshold is configured to 1 the first client will be served by multicast.
- If multicastThreshold is configured to 2 the first client will be served with unicast and as soon as the second client arrives both will be served by multicast, etc.

[TPS_MANI_03020]{DRAFT} **Servers RequestResponseDelay for received SubscribeEventGroup entries** ⌈ The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SomeipSdClientEventGroupTimingConfig.requestResponseDelay.

The actual delay will be randomly chosen between the maxValue and minValue. ⌋ *(RS_MANI_00024)*

Class	**SomeipSdServerEventGroupTimingConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	EventGroup specific timing configuration settings. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
request ResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service). **Tags:** atp.Status=draft

Table 9.37: SomeipSdServerEventGroupTimingConfig

9.2.1.1.5 ProvidedSomeipServiceInstance related event and method properties

[TPS_MANI_03154]{DRAFT} **ProvidedSomeipServiceInstance related configuration settings for events** ⌈ The class SomeipEventProps that is aggregated by the ProvidedSomeipServiceInstance in the role eventProps allows to specify ProvidedSomeipServiceInstance related configuration settings for events that are defined in the SomeipServiceInterfaceDeployment referenced by the ProvidedSomeipServiceInstance in the role serviceInterface. ⌋ *(RS_MANI_00024)*

[TPS_MANI_03155]{DRAFT} **ProvidedSomeipServiceInstance related configuration settings for methods** ⌈ The class SomeipMethodProps that is aggregated by the ProvidedSomeipServiceInstance in the role methodResponseProps allows to specify ProvidedSomeipServiceInstance related configuration settings for a method response message. The method is defined in the SomeipServiceInterfaceDeployment referenced by the ProvidedSomeipServiceInstance in the role serviceInterface. ⌋*(RS_MANI_00024)*

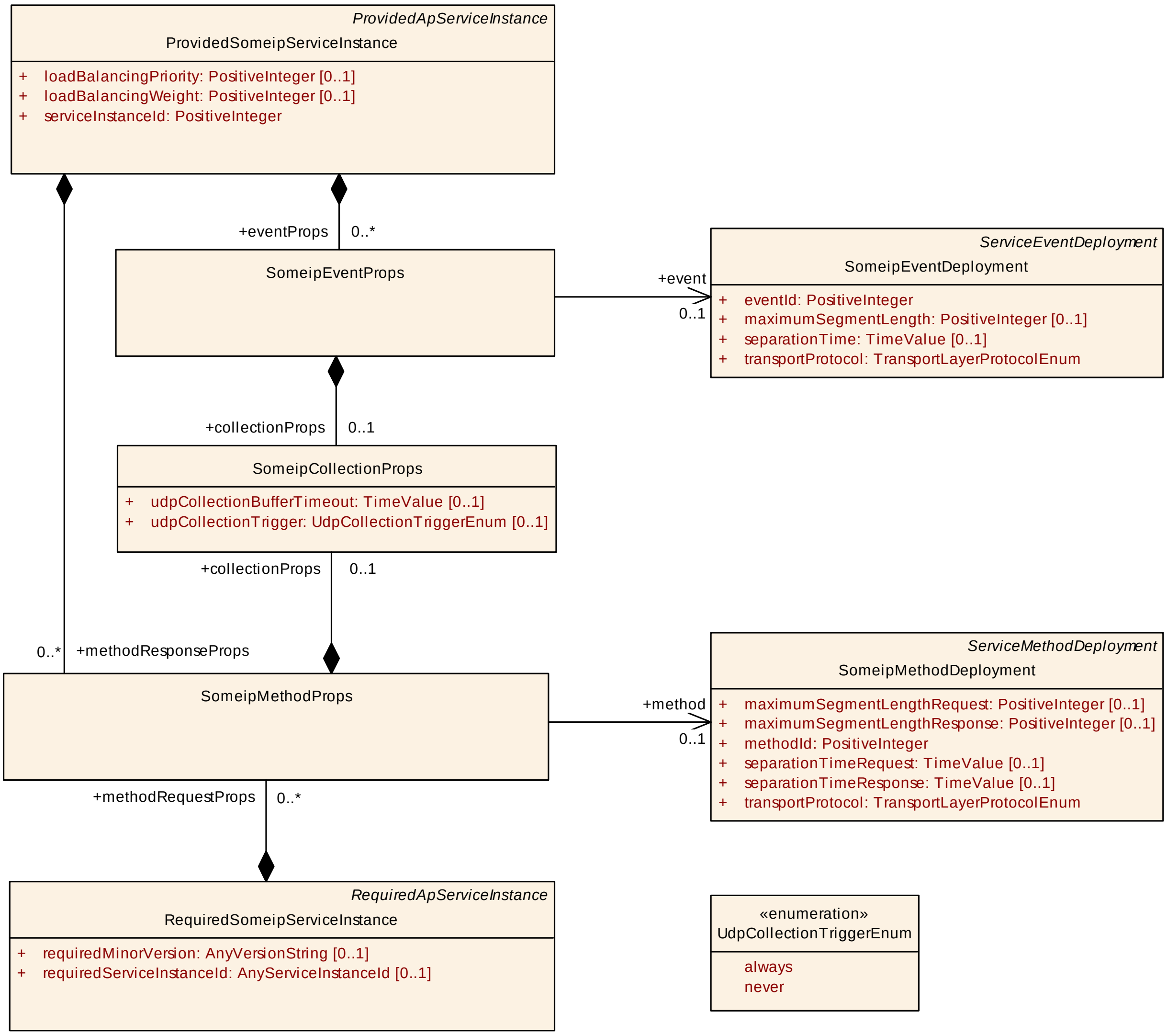


Figure 9.13: ProvidedSomeipServiceInstance related event and method properties

Class	**SomeipEventProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class allows to set configuration options for an event in the provided service instance. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
collectionProps	SomeipCollectionProps	0..1	aggr	Collection of timing attributes configurable for an event that is provided by a Service Instance. **Tags:** atp.Status=draft

▽

△

Class	**SomeipEventProps**			
event	SomeipEvent Deployment	0..1	ref	Reference to the event for which the SomeipEventProps are applicable. **Tags:** atp.Status=draft

Table 9.38: SomeipEventProps

Class	**SomeipMethodProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class allows to set configuration options for a method in the service instance. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
collectionProps	SomeipCollectionProps	0..1	aggr	Collection of timing attributes configurable for a method that is provided or requested by a Service Instance. **Tags:** atp.Status=draft
method	SomeipMethod Deployment	0..1	ref	Reference to the method for which the SomeipMethod Props are applicable. **Tags:** atp.Status=draft

Table 9.39: SomeipMethodProps

Class	**SomeipCollectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Collection of attributes that are configurable for an event that is provided by a ServiceInstance or for a method that is provided or requested by a ServiceInstance. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
udpCollection BufferTimeout	TimeValue	0..1	attr	Maximum time, an outgoing message (event, method call or method response) may be delayed, due to data collection.
udpCollection Trigger	UdpCollectionTrigger Enum	0..1	attr	Defines whether the ServiceInterface element (event or method) contributes to the triggering of the udp data transmission if data collection is enabled.

Table 9.40: SomeipCollectionProps

Enumeration	**UdpCollectionTriggerEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	Defines whether the ServiceInterface element (event or method) contributes to the triggering of the udp data transmission if data collection is enabled. **Tags:** atp.Status=draft
Literal	*Description*

▽

△

Enumeration	**UdpCollectionTriggerEnum**
always	ServiceInterface element will trigger the transmission of the data. **Tags:** atp.EnumerationValue=0
never	ServiceInterface element will be buffered and will not trigger the transmission of the data. **Tags:** atp.EnumerationValue=1

Table 9.41: UdpCollectionTriggerEnum

[TPS_MANI_03158]{DRAFT} **Configuration of a data collection on a ProvidedServiceInstance for transmission over udp** ⌈ The attributes udpCollectionBufferTimeout and udpCollectionTrigger support the configuration of a data collection of several messages for transmission over udp. In the ProvidedServiceInstance all method responses and events for which the udpCollectionTrigger is set to never will be collected in a buffer until a trigger arrives that starts the data transmission.

The following trigger options are supported:

- a message needs to be transmitted for which the udpCollectionTrigger is set to always.
- the udpCollectionBufferTimeout is reached for a message.
- the buffer size defined by the attribute udpCollectionBufferSizeThreshold is reached.

⌋*(RS_MANI_00024)*

9.2.1.2 Required Service Instance

[TPS_MANI_03059]{DRAFT} **RequiredSomeipServiceInstance.requiredServiceInstanceId** ⌈ The RequiredSomeipServiceInstance defines the requiredServiceInstanceId of a SomeipServiceInterfaceDeployment that the client searches.

The client may search for a specific requiredServiceInstanceId or for ANY requiredServiceInstanceId of the serviceInterface. ⌋*(RS_MANI_00024)*

Class	**RequiredSomeipServiceInstance**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstances

▽

△

Class	RequiredSomeipServiceInstance			
Base	*ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, RequiredApServiceInstance, Uploadable PackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
capability Record (ordered)	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Tags:** atp.Status=draft
methodRequest Props	SomeipMethodProps	*	aggr	Configuration settings for individual methods that are requested by the ServiceInstance. **Tags:** atp.Status=draft
requiredEvent Group	SomeipRequiredEvent Group	*	aggr	List of EventGroups that are used by the RequiredService Instance. **Tags:** atp.Status=draft
requiredMinor Version	AnyVersionString	0..1	attr	This attribute is used to configure for which minor version of the SomeIp ServiceInterface the Service Discovery will search. Value can be set to a number that represents the Minor Version of the searched service or to ANY.
requiredService InstanceId	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID.
sdClientConfig	SomeipSdClientService InstanceConfig	1	ref	Client specific configuration settings relevant for the SOME/IP service discovery. **Tags:** atp.Status=draft

Table 9.42: RequiredSomeipServiceInstance

[TPS_MANI_03021]{DRAFT} **Requirements on the searched minor version from the client's point of view** ⌈ The meta-class `RequiredSomeipServiceInstance` is able to make further specifications regarding the version of the service from the client's point of view.

For this purpose, the attribute `RequiredSomeipServiceInstance.requiredMinorVersion` exists and provides the ability to define the required minor version (`SomeipServiceInterfaceVersion.minorVersion`). ⌋*(RS_MANI_00024)*

Please note that the major version that the client searches for is already defined by the `SomeipServiceInterfaceVersion.majorVersion` in the `SomeipServiceInterfaceDeployment`. It is therefore not possible to search for ANY major version, so the client looks always for a specific major version.

The minor version that may be defined by `SomeipServiceInterfaceVersion.minorVersion` in the `SomeipServiceInterfaceDeployment` is irrelevant for the client and the service search and shall be ignored.

9.2.1.2.1 IP Configuration

In SOME/IP, the SubscribeEventGroup entry references IPv4 or IPv6 Endpoint options to indicate to the server where the client wants to receive the events of the SomeipEventGroup. Such an Endpoint contains the IP address of the client.

[TPS_MANI_03022]{DRAFT} **Context of RequiredSomeipServiceInstance** ⌈ A RequiredSomeipServiceInstance can be mapped to a CommunicationConnector of a MachineDesign with the SomeipServiceInstanceToMachineMapping.

With this mapping an assignment of the RequiredSomeipServiceInstance to a unicast IP Address is established since the EthernetCommunicationConnector refers to a NetworkEndpoint in the role unicastNetworkEndpoint. The unicastNetworkEndpoint defines the local IP address of the client. ⌋ *(RS_MANI_00009, RS_MANI_00024)*

[constr_3411]{DRAFT} **eventMulticastUdpPort, ipv4MulticastIpAddress and ipv6MulticastIpAddress not relevant for RequiredSomeipServiceInstances** ⌈ The attributes eventMulticastUdpPort, ipv4MulticastIpAddress and ipv6MulticastIpAddress are not relevant for a ServiceInstanceToMachineMapping that maps a RequiredSomeipServiceInstance to a CommunicationConnector of a MachineDesign. ⌋ *()*

The reason for [constr_3411] is that the Server informs the client in the Subscribe Eventgroup Acknowledgement entry to which Multicast IP address the event messages will be send. There is no reason to configure the IPv4 Multicast Option or IPv6 Multicast Option on the client side.

9.2.1.2.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP SubscribeEventGroup message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the client.

With the SomeipServiceInstanceToMachineMapping the Transport Layer configuration attributes are assigned to the RequiredSomeipServiceInstance.

The Transport Layer (TCP/UDP) configuration attributes for the SubscribeEventGroup entry are directly available in the SomeipServiceInstanceToMachineMapping element.

The SomeipServiceInstanceToMachineMapping defines also the source-port where the client sends the method call messages to the server and the destination-port where the client receives the method responses from the server.

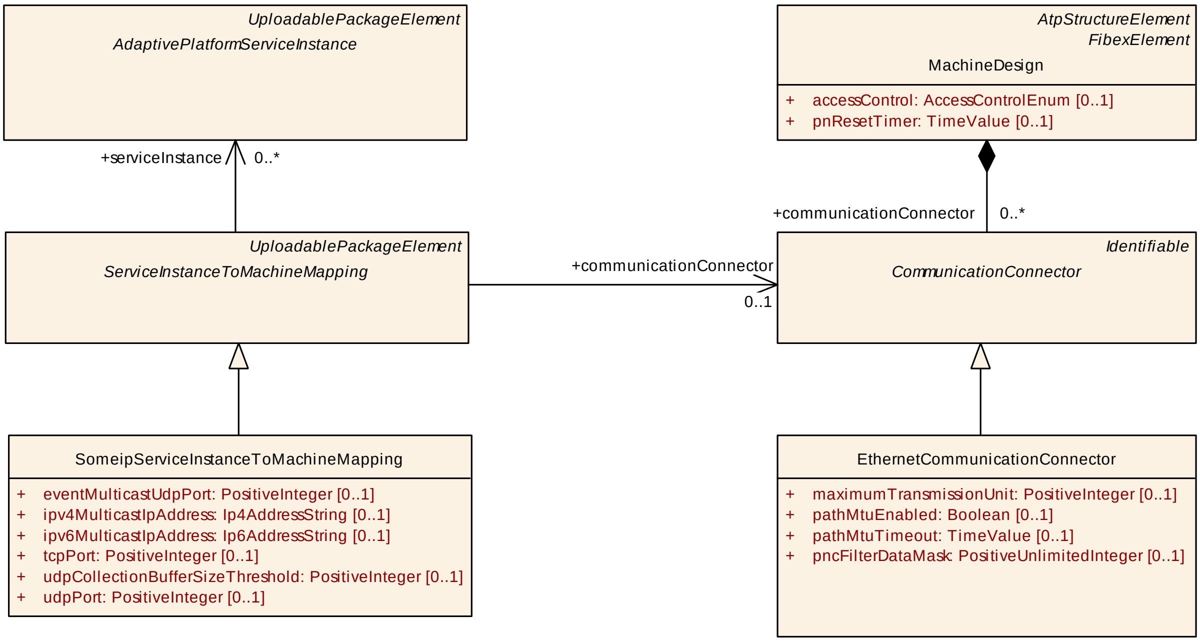


Figure 9.14: SomeipServiceInstanceToMachineMapping with TP and IP configuration

[TPS_MANI_03023]{DRAFT} **Udp Transport Protocol Configuration for RequiredSomeipServiceInstance** ⌈ The SomeipServiceInstanceToMachineMapping.udpPort defines the Transport Protocol for a UDP communication in case that the server provides ServiceInterface content over UDP and the client wants to use it. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03024]{DRAFT} **Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance** ⌈ The SomeipServiceInstanceToMachineMapping.tcpPort defines the Transport Protocol for a TCP communication in case that the server provides ServiceInterface content over TCP and the client wants to use it. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03049]{DRAFT} **Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance** ⌈ It is allowed to set tcpPort and udpPort in the same SomeipServiceInstanceToMachineMapping. Such a setting shall be used in case that the server provides ServiceInterface content over Udp and Tcp and the client wants to use it. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3296]{DRAFT} **Transport Protocol attributes defined for a RequiredSomeipServiceInstance** ⌈ Each SomeipServiceInstanceToMachineMapping that is defined for a RequiredSomeipServiceInstance shall define either

- a udpPort or
- a tcpPort or
- a udpPort and a tcpPort.

⌋*()*

If a Tcp and Udp Transport Protocol Configuration is defined for a RequiredSomeipServiceInstance as described in [TPS_MANI_03049] then the SOME/IP ServiceInterfaceDeployment settings decide which content of the ProvidedSomeipServiceInstance is transported over udp and which content is transported over tcp. This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

9.2.1.2.3 Service Discovery Client Configuration

Service Discovery phases on the Client side allow minimizing the number of Service Discovery messages and allow a fast synchronization upon ECU start.

For every RequiredSomeipServiceInstance on a Client different phases are existing:

- Down
- Requested
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase

[TPS_MANI_03025]{DRAFT} **Client Timing configuration for a RequiredSomeipServiceInstance** ⌈ The Client Timing is configurable with SomeipSdClientServiceInstanceConfig that is referenced in the role sdClientConfig by the RequiredSomeipServiceInstance for which the Timing is valid. ⌋ *(RS_MANI_00024)*

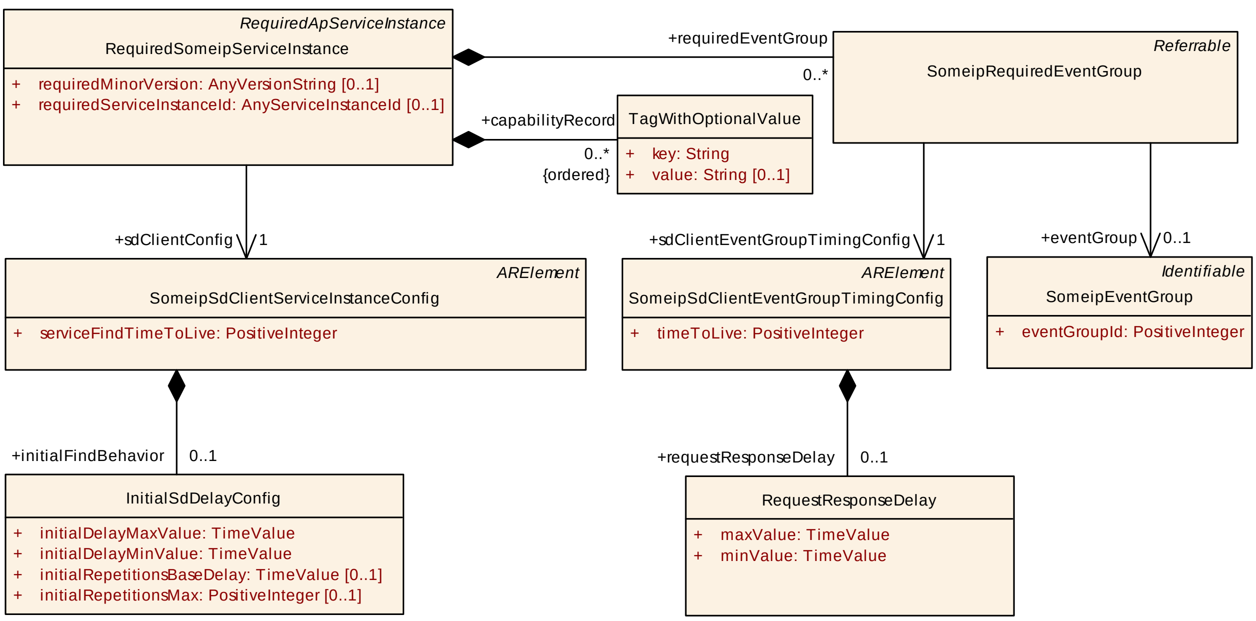


Figure 9.15: SOME/IP Service Discovery Client configuration settings

Class	SomeipSdClientServiceInstanceConfig			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Client specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
initialFind Behavior	InitialSdDelayConfig	0..1	aggr	Controls initial find behavior of clients. **Tags:** atp.Status=draft
serviceFind TimeToLive	PositiveInteger	1	attr	This attribute represents the ability to define the time in seconds the service find is valid.

Table 9.43: SomeipSdClientServiceInstanceConfig

[TPS_MANI_03026]{DRAFT} **Initial Wait Phase configuration for a RequiredSomeipServiceInstance** ⌈ The Initial Wait Phase for a RequiredSomeipServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue.

If a calculated random timer based on these min and max values expires the first FindService entry will be sent out. ⌋*(RS_MANI_00024)*

When the calculated random timer expires and no OfferService is received the Repetition Phase will be entered.

[TPS_MANI_03027]{DRAFT} **Repetition Wait Phase configuration for a RequiredSomeipServiceInstance** ⌈ The Repetition Wait Phase for a RequiredSomeipServiceInstance is configured with the initialFindBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay. ⌋*(RS_MANI_00024)*

If the Repetition Phase is entered, the Service Discovery waits the initialRepetitionsBaseDelay and sends an FindService entry.

If the amount of sent FindService entries reaches initialRepetitionsMax and no OfferService is received the Main Phase will be entered. In the Main Phase no further FindService entries are send by the client.

[TPS_MANI_03028]{DRAFT} **TTL for Find Service Entries** ⌈ The lifetime of a RequiredSomeipServiceInstance is configurable with the serviceFindTimeToLive attribute of SomeipSdClientServiceInstanceConfig.

If the time that is configured by serviceFindTimeToLive expires the FindService entry shall be considered not existing. ⌋*(RS_MANI_00024)*

Figure 9.16 shows an example of the different SOME/IP phases on the Client side.

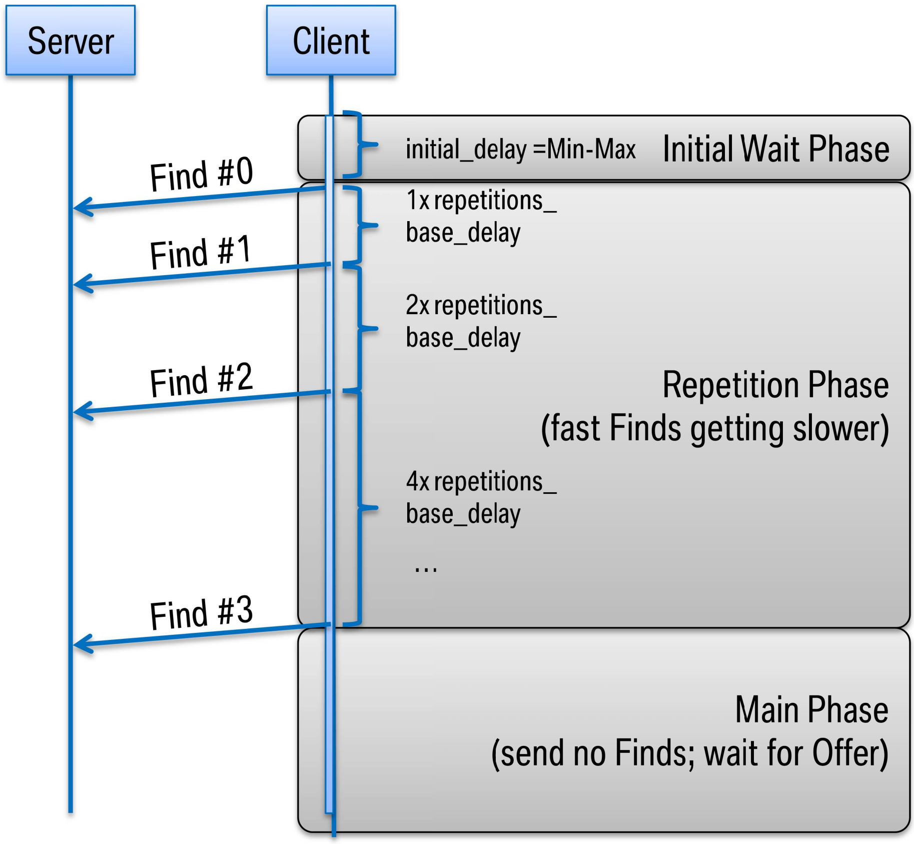


Figure 9.16: SOME/IP Client Timing example

SOME/IP allows to specify additional information about the RequiredSomeipServiceInstance with the Capability Record that allows to transport arbitrary configuration strings (key/value pairs).

This allows to encode additional information like the name of a service or its configuration.

[TPS_MANI_03029]{DRAFT} **Client Capability Records** ⌈ A Capability Record (key/-value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value. ⌋*(RS_MANI_00024)*

9.2.1.2.4 Required Event Group

The RequiredSomeipServiceInstance aggregates a SomeipRequiredEventGroup in the role requiredEventGroup that allows to define service instance specific configuration settings for a SomeipEventGroup.

Class	**SomeipRequiredEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the required side for each EventGroup separately. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related Event Group settings are valid. **Tags:** atp.Status=draft
sdClientEventGroupTimingConfig	SomeipSdClientEventGroupTimingConfig	1	ref	Client Timing configuration settings that are EventGroup specific. **Tags:** atp.Status=draft

Table 9.44: SomeipRequiredEventGroup

Class	**SomeipSdClientEventGroupTimingConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class is used to specify configuration related to service discovery in the context of an event group on SOME/IP. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
requestResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service). **Tags:** atp.Status=draft
timeToLive	PositiveInteger	1	attr	Defines the time in seconds the subscription of this event is expected by the client. this value is sent from the client to the server in the SD-subscribeEvent message.

Table 9.45: SomeipSdClientEventGroupTimingConfig

[TPS_MANI_03030]{DRAFT} **SomeipSdClientEventGroupTimingConfig.timeToLive for SubscribeEventGroup Entries** ⌈ The lifetime of a event subscription is configurable with the timeToLive attribute of SomeipSdClientEventGroupTimingConfig.

If the time that is configured by timeToLive expires the event subscription is canceled. ⌋*(RS_MANI_00024)*

[TPS_MANI_03031]{DRAFT} **Clients RequestResponseDelay for received ServiceOffer entries** ⌈ The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SomeipSdClientEventGroupTimingConfig.requestResponseDelay.

The actual delay will be randomly chosen between the maxValue and minValue. ⌋
(RS_MANI_00024)

9.2.1.2.5 RequiredSomeipServiceInstance related method call properties

[TPS_MANI_03156]{DRAFT} **RequiredSomeipServiceInstance related configuration settings for methods** ⌈ The class SomeipMethodProps that is aggregated by the RequiredSomeipServiceInstance in the role methodRequestProps allows to specify RequiredSomeipServiceInstance related configuration settings for a method request message. The method is defined in the SomeipServiceInterfaceDeployment referenced by the RequiredSomeipServiceInstance in the role serviceInterface. ⌋*(RS_MANI_00024)*

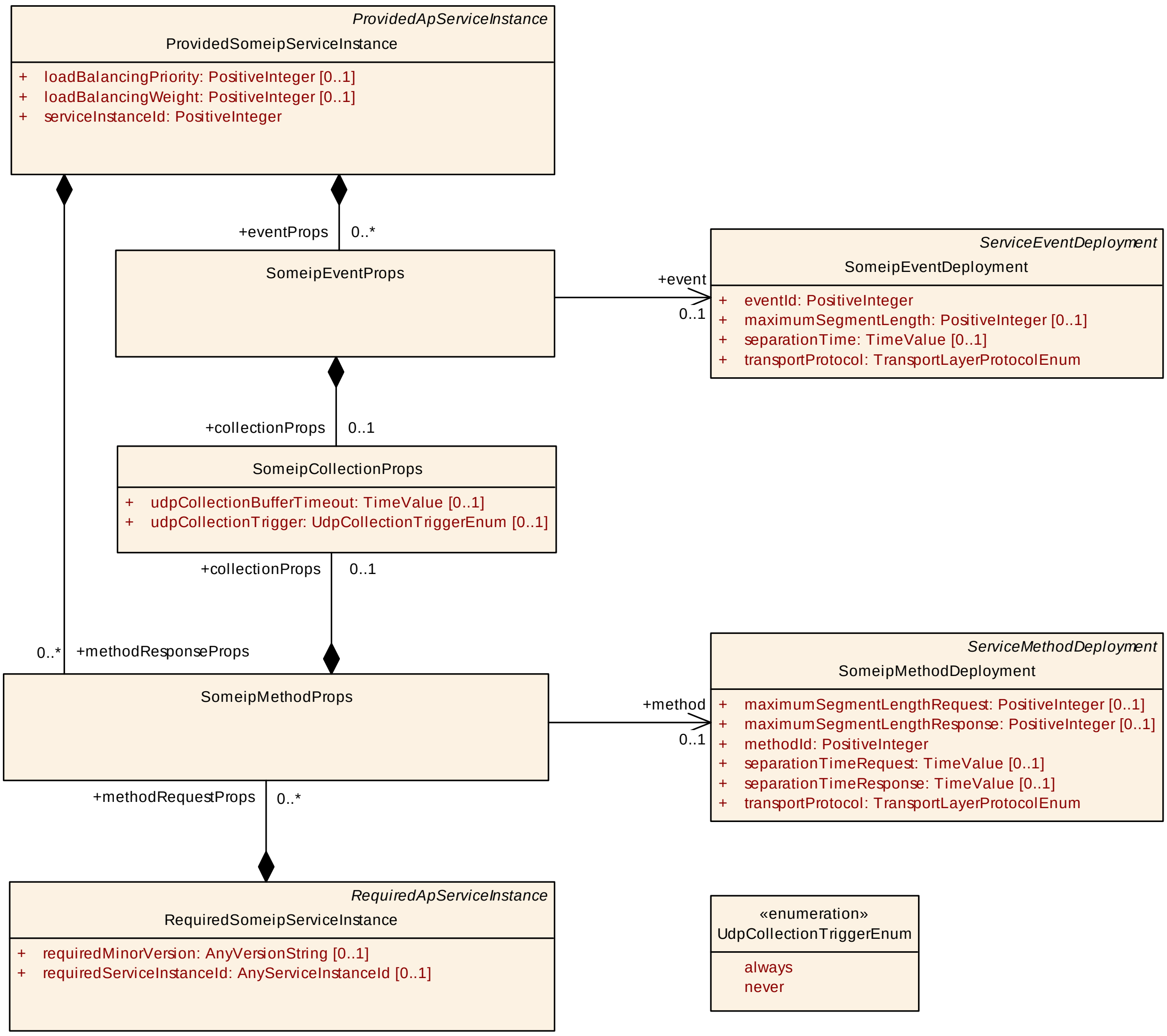


Figure 9.17: RequiredSomeipServiceInstance related event and method properties

[TPS_MANI_03159]{DRAFT} **Configuration of a data collection on a RequiredSomeipServiceInstance for transmission over udp** ⌈ The attributes udpCollectionBufferTimeout and udpCollectionTrigger support the configuration of a data collection of several messages for transmission over udp. In the RequiredSomeipServiceInstance all method requests for which the udpCollectionTrigger is set to never will be collected in a buffer until a trigger arrives that starts the data transmission.

The following trigger options are supported:

- a message needs to be transmitted for which the udpCollectionTrigger is set to always.
- the udpCollectionBufferTimeout is reached for a message.
- the buffer size defined by the attribute udpCollectionBufferSizeThreshold is reached.

⌋*(RS_MANI_00024)*

9.2.2 DDS Service Instance Deployment

In the case of DDS used as the transport layer the derived meta-classes are DdsProvidedServiceInstance or DdsRequiredServiceInstance. These meta-classes also carry attributes that apply for the service discovery on DDS.

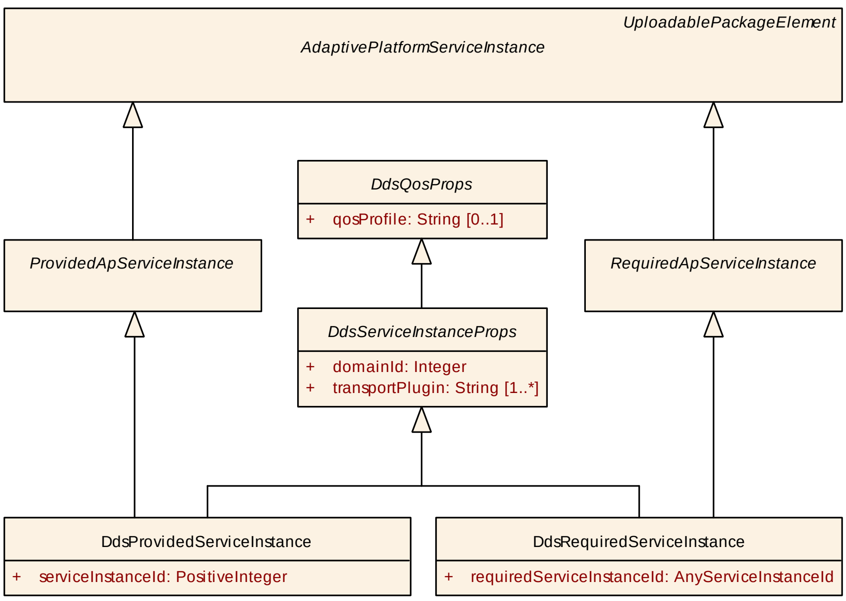


Figure 9.18: Dds Service Instances

Class	***DdsQosProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	QoS configuration properties for the DDS entities associated with an event, method, or field provided by or requested from a Service Instance using DDS as the underlying network binding. **Tags:** atp.Status=draft			
Base	*ARObject*			
Subclasses	DdsEventQosProps, DdsFieldQosProps, DdsMethodQosProps, *DdsServiceInstanceProps*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
qosProfile	String	0..1	attr	Identifies a group of QoS Policies that apply to the DDS entities associated with the event, method, field, or the service instance. **Tags:** atp.Status=draft

Table 9.46: DdsQosProps

Class	***DdsServiceInstanceProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Common configuration properties for the DDS entities provided by or requested from a Service Instance using DDS as the underlying network binding. **Tags:** atp.Status=draft			
Base	*ARObject*, *DdsQosProps*			
Subclasses	DdsProvidedServiceInstance, DdsRequiredServiceInstance			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
domainId	Integer	1	attr	This attribute identifies the DDS Domain the Service Instance shall join. **Tags:** atp.Status=draft
transportPlugin	String	1..*	attr	Enable a transport plug-in (e.g., sharedMemory) in the underlying DDS binding implementation. **Tags:** atp.Status=draft

Table 9.47: DdsServiceInstanceProps

9.2.2.1 Provided DDS Service Instance

[TPS_MANI_03527]{DRAFT} **Definition of `DdsProvidedServiceInstance`** ⌈The `DdsProvidedServiceInstance` configures the Service to join a DDS Domain with the `domainId` attribute, and to instantiate the underlying DDS entities according to a QoS Profile with the `qosProfile` attribute. Moreover, it assigns an Instance Id to the Service for deployment with the `serviceInstanceId` attribute. ⌋*(RS_MANI_00038)*

[constr_3528]{DRAFT} **Value range of `domainId`** ⌈The value of `domainId` at `DdsProvidedServiceInstance` and `domainId` at `DdsRequiredServiceInstance` shall be in the range of a signed 32 bit integer. ⌋*()*

[constr_3529]{DRAFT} **Value range of `serviceInstanceId`** ⌈The value of `serviceInstanceId` shall be in the range of 0..65535. ⌋*()*

[constr_3541]{DRAFT} **`qosProfile` mandatory for `DdsProvidedServiceInstance`** ⌈ The attribute `qosProfile` shall be defined for every `DdsProvidedServiceInstance`. ⌋*()*

[TPS_MANI_03571]{DRAFT} **`transportPlugin` for `DdsProvidedServiceInstance`** ⌈ The attribute `transportPlugin` specifies the list of transport plugins of the underlying DDS implementation that shall be enabled on the server side upon instantiation, so that the proxy can communicate with remote applications through a specific transport-level protocol or communication technology. A service will only be able to interact with proxies if a common transport plugin is enabled on the remote end.

This specification standardizes the following values for `transportPlugin` but further values may be used by specific implementations:

- `sharedMemory`
- `udp`
- `tcp`

⌋*(RS_MANI_00038)*

[constr_3543]{DRAFT} **At least one `transportPlugin` definition required for each `DdsProvidedServiceInstance`** ⌈ At least one `transportPlugin` shall be defined for each `DdsProvidedServiceInstance`. ⌋*()*

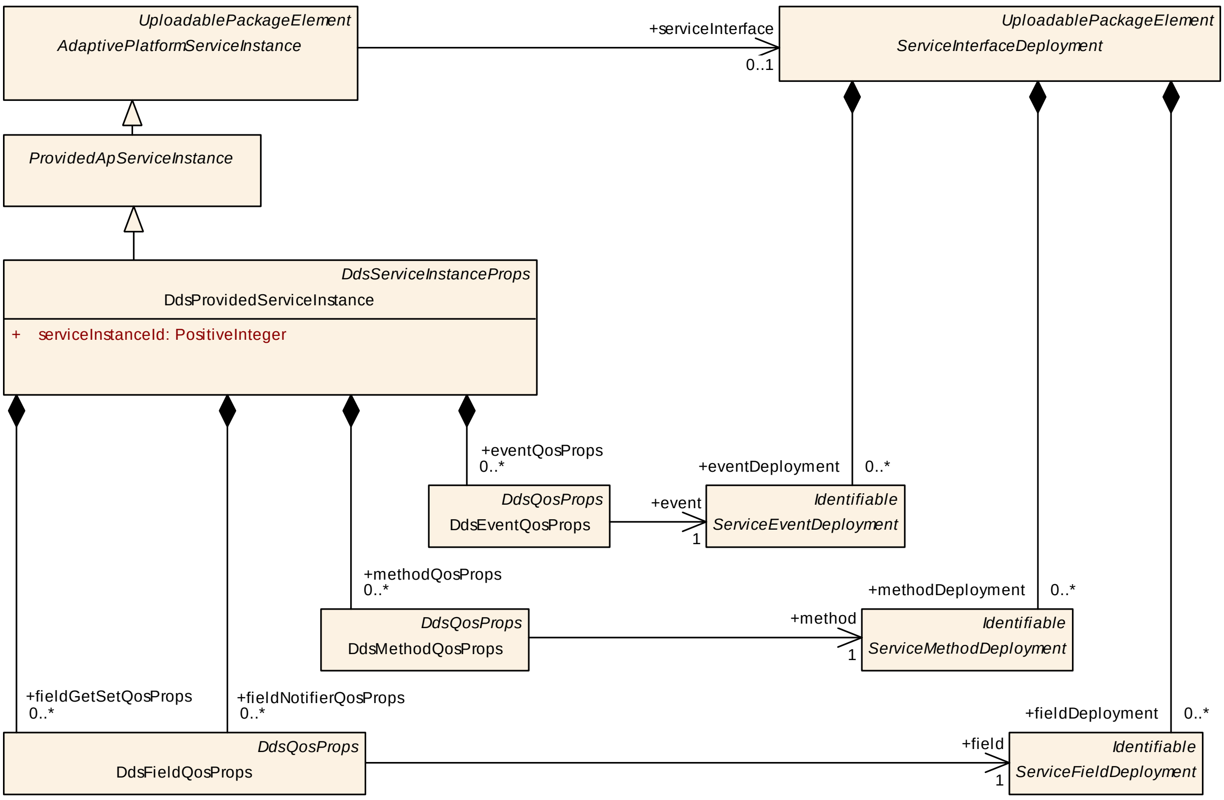


Figure 9.19: Provided Dds Service Instances

Class	DdsProvidedServiceInstance			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of DDS. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *DdsQosProps*, *DdsServiceInstanceProps*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventQosProps	DdsEventQosProps	*	aggr	List of configuration properties for the Events that are provided by the Service Instance. **Tags:** atp.Status=draft
fieldGetSetQosProps	DdsFieldQosProps	*	aggr	List of configuration properties for the DDS-RPC service that provides access to the field getters/setters of the service instance. **Tags:** atp.Status=draft
fieldNotifierQosProps	DdsFieldQosProps	*	aggr	List of configuration properties for Field notifiers that are provided by the Service Instance. **Tags:** atp.Status=draft
methodQosProps	DdsMethodQosProps	*	aggr	List of configuration properties for the DDS-RPC service that provides the methods of the Service Instance. **Tags:** atp.Status=draft
serviceInstanceId	PositiveInteger	1	attr	Identification number that is used by DDS to identify DomainParticipants associated with an instance of the service. **Tags:** atp.Status=draft

Table 9.48: DdsProvidedServiceInstance

[TPS_MANI_03528]{DRAFT} **Definition of `DdsProvidedServiceInstance.eventQosProps`** ⌈ The `DdsProvidedServiceInstance.eventQosProps` configures the DDS entities associated with the `event` according to a QoS Profile specified with the `qosProfile` attribute. ⌋*(RS_MANI_00038)*

[TPS_MANI_03531]{DRAFT} **`qosProfile` of `DdsProvidedServiceInstance.eventQosProps` is optional** ⌈ The attribute `qosProfile` of `DdsProvidedServiceInstance.eventQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsProvidedServiceInstance`. ⌋*(RS_MANI_00038)*

Class	DdsEventQosProps
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	Configuration properties of the Event using DDS as the underlying network binding. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft
Base	*ARObject*, *DdsQosProps*

▽

△

Class	**DdsEventQosProps**			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
event	ServiceEvent Deployment	1	ref	Reference to an event that is provided. **Tags:** atp.Status=draft

Table 9.49: DdsEventQosProps

[TPS_MANI_03559]{DRAFT} **Definition of `DdsProvidedServiceInstance.methodQosProps`** ⌈ The `DdsProvidedServiceInstance.methodQosProps` configures the DDS entities associated with the `method` according to a QoS Profile specified with the `qosProfile` attribute. ⌋*(RS_MANI_00038)*

[TPS_MANI_03560]{DRAFT} **`qosProfile` of `DdsProvidedServiceInstance.methodQosProps` is optional** ⌈ The attribute `qosProfile` of `DdsProvidedServiceInstance.methodQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsProvidedServiceInstance`. ⌋*(RS_MANI_00038)*

Class	**DdsMethodQosProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Configuration properties of the Method that handles method request/replies when using DDS as the underlying network binding. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *DdsQosProps*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
method	ServiceMethod Deployment	1	ref	Reference to the method. **Tags:** atp.Status=draft

Table 9.50: DdsMethodQosProps

[TPS_MANI_03561]{DRAFT} **Definition of `DdsProvidedServiceInstance.fieldNotifierQosProps`** ⌈ The `DdsProvidedServiceInstance.fieldNotifierQosProps` configures the DDS entities associated with the `field` according to a QoS Profile specified with the `qosProfile` attribute. ⌋*(RS_MANI_00038)*

[TPS_MANI_03562]{DRAFT} **`qosProfile` of `DdsProvidedServiceInstance.fieldNotifierQosProps` is optional** ⌈ The attribute `qosProfile` of `DdsProvidedServiceInstance.fieldNotifierQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsProvidedServiceInstance`. ⌋*(RS_MANI_00038)*

[TPS_MANI_03563]{DRAFT} **Definition of `DdsProvidedServiceInstance.fieldGetSetQosProps`** ⌈ The `DdsProvidedServiceInstance.fieldGetSetQosProps` configures the DDS entities associated with the `field` according to a QoS Profile specified with the `qosProfile` attribute. ⌋*(RS_MANI_00038)*

[TPS_MANI_03564]{DRAFT} **qosProfile of DdsProvidedServiceInstance.fieldGetSetQosProps is optional** ⌈ The attribute qosProfile of DdsProvidedServiceInstance.fieldGetSetQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsProvidedServiceInstance. ⌋ *(RS_MANI_00038)*

Class	DdsFieldQosProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Configuration properties of the Field interaction when using DDS as the underlying network binding. **Tags:** atp.Status=draft			
Base	*ARObject*, *DdsQosProps*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
field	ServiceField Deployment	1	ref	Reference to the field. **Tags:** atp.Status=draft

Table 9.51: DdsFieldQosProps

9.2.2.2 Required DDS Service Instance

[TPS_MANI_03529]{DRAFT} **Definition of DdsRequiredServiceInstance** ⌈ The DdsRequiredServiceInstance configures the Client to join a DDS Domain with the domainId attribute, and to instantiate the underlying DDS entities according to a QoS Profile with the qosProfile attribute. Optionally, the requiredServiceInstanceId attribute allows a Client to search for a specific Instance Id of the serviceInterface. ⌋ *(RS_MANI_00038)*

[constr_3542]{DRAFT} **qosProfile mandatory for DdsRequiredServiceInstance** ⌈ The attribute qosProfile shall be defined for every DdsRequiredServiceInstance. ⌋ *()*

[TPS_MANI_03572]{DRAFT} **transportPlugin for DdsRequiredServiceInstance** ⌈ The attribute transportPlugin specifies the list of transport plugins of the underlying DDS implementation that shall be enabled on the client side upon instantiation, so that the proxy can communicate with remote applications through a specific transport-level protocol or communication technology. A service will only be able to interact with proxies if a common transport plugin is enabled on the remote end.

This specification standardizes the following values for transportPlugin but further values may be used by specific implementations:

- sharedMemory
- udp
- tcp

⌋ *(RS_MANI_00038)*

[constr_3544]{DRAFT} **At least one transportPlugin definition required for each DdsRequiredServiceInstance** ⌈ At least one transportPlugin shall be defined for each DdsRequiredServiceInstance. ⌋*()*

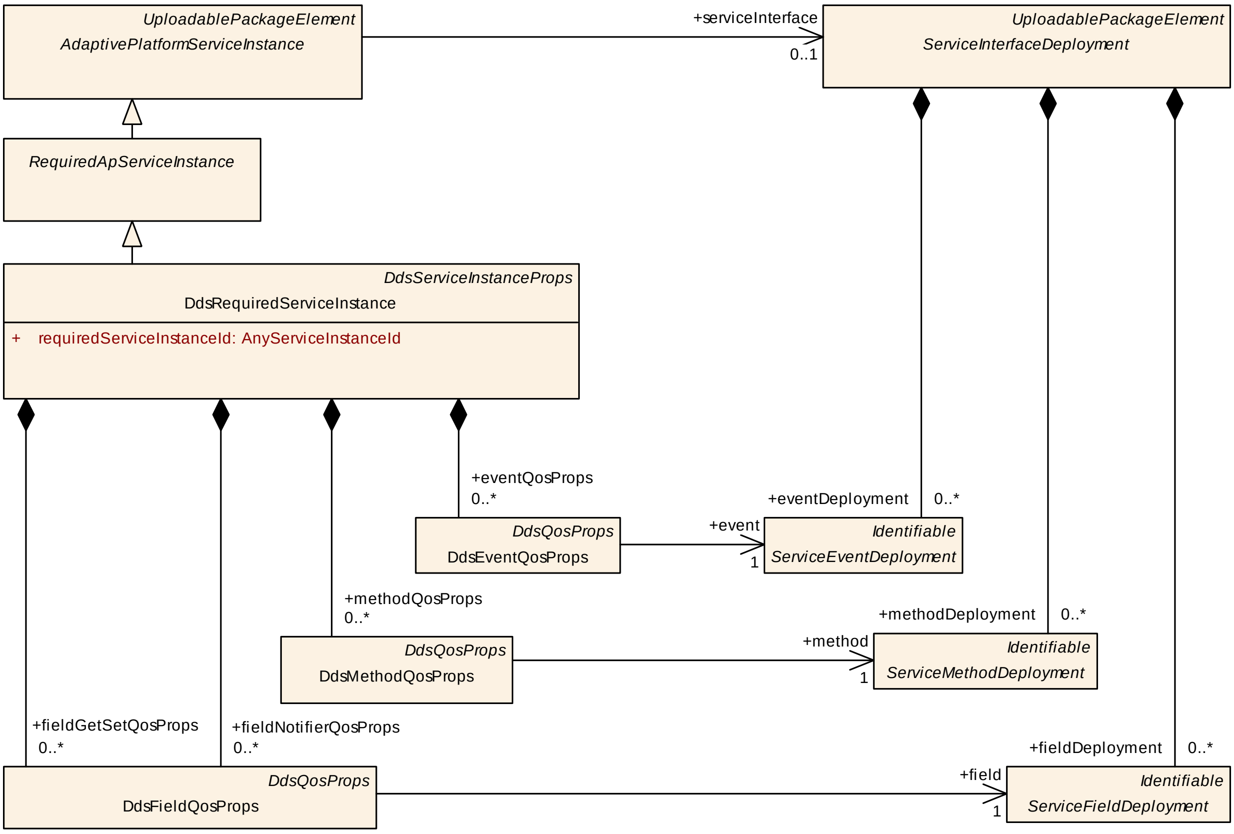


Figure 9.20: Required Dds Service Instances

Class	**DdsRequiredServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of DDS. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *DdsQosProps*, *DdsServiceInstanceProps*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventQosProps	DdsEventQosProps	*	aggr	List of configuration properties for the Events that are required by the Service Instance. **Tags:** atp.Status=draft
fieldGetSetQosProps	DdsFieldQosProps	*	aggr	List of configuration properties for the DDS-RPC service that requires access to the field getters/setters of the service instance. **Tags:** atp.Status=draft

▽

△

Class	**DdsRequiredServiceInstance**			
fieldNotifierQos Props	DdsFieldQosProps	*	aggr	List of configuration properties for Field notifiers that are required by the Service Instance. **Tags:** atp.Status=draft
methodQos Props	DdsMethodQosProps	*	aggr	List of configuration properties for the DDS-RPC service that requires access to the methods of the service instance. **Tags:** atp.Status=draft
requiredService InstanceId	AnyServiceInstanceId	1	attr	This attribute represents the ability to describe the required service instance ID. **Tags:** atp.Status=draft

Table 9.52: DdsRequiredServiceInstance

[TPS_MANI_03530]{DRAFT} **Definition of DdsRequiredServiceInstance.eventQosProps** ⌈ The DdsRequiredServiceInstance.eventQosProps configures the DDS entities responsible for subscribing to an event according to a QoS Profile specified with the qosProfile attribute. ⌋*(RS_MANI_00038)*

[TPS_MANI_03532]{DRAFT} **qosProfile of DdsRequiredServiceInstance.eventQosProps is optional** ⌈ The attribute qosProfile of DdsRequiredServiceInstance.eventQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsRequiredServiceInstance. ⌋*(RS_MANI_00038)*

[TPS_MANI_03565]{DRAFT} **Definition of DdsRequiredServiceInstance.methodQosProps** ⌈ The DdsRequiredServiceInstance.methodQosProps configures the DDS entities associated with the method according to a QoS Profile specified with the qosProfile attribute. ⌋*(RS_MANI_00038)*

[TPS_MANI_03566]{DRAFT} **qosProfile of DdsRequiredServiceInstance.methodQosProps is optional** ⌈ The attribute qosProfile of DdsRequiredServiceInstance.methodQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsRequiredServiceInstance. ⌋*(RS_MANI_00038)*

[TPS_MANI_03567]{DRAFT} **Definition of DdsRequiredServiceInstance.fieldNotifierQosProps** ⌈ The DdsRequiredServiceInstance.fieldNotifierQosProps configures the DDS entities associated with the field according to a QoS Profile specified with the qosProfile attribute. ⌋*(RS_MANI_00038)*

[TPS_MANI_03568]{DRAFT} **qosProfile of DdsRequiredServiceInstance.fieldNotifierQosProps is optional** ⌈ The attribute qosProfile of DdsRequiredServiceInstance.fieldNotifierQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsRequiredServiceInstance. ⌋*(RS_MANI_00038)*

[TPS_MANI_03569]{DRAFT} **Definition of DdsRequiredServiceInstance.fieldGetSetQosProps** ⌈ The DdsRequiredServiceInstance.fieldGetSetQosProps configures the DDS entities associated with the field according to a QoS Profile specified with the qosProfile attribute. ⌋*(RS_MANI_00038)*

[TPS_MANI_03570]{DRAFT} **qosProfile of DdsRequiredServiceInstance.fieldGetSetQosProps is optional** ⌈ The attribute qosProfile of DdsRequiredServiceInstance.fieldGetSetQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsRequiredServiceInstance. ⌋ *(RS_MANI_00038)*

9.2.2.3 DDS Service Instance to Machine mapping

The DdsServiceInstanceToMachineMapping defines on which network / VLAN the DDS communication shall be deployed.

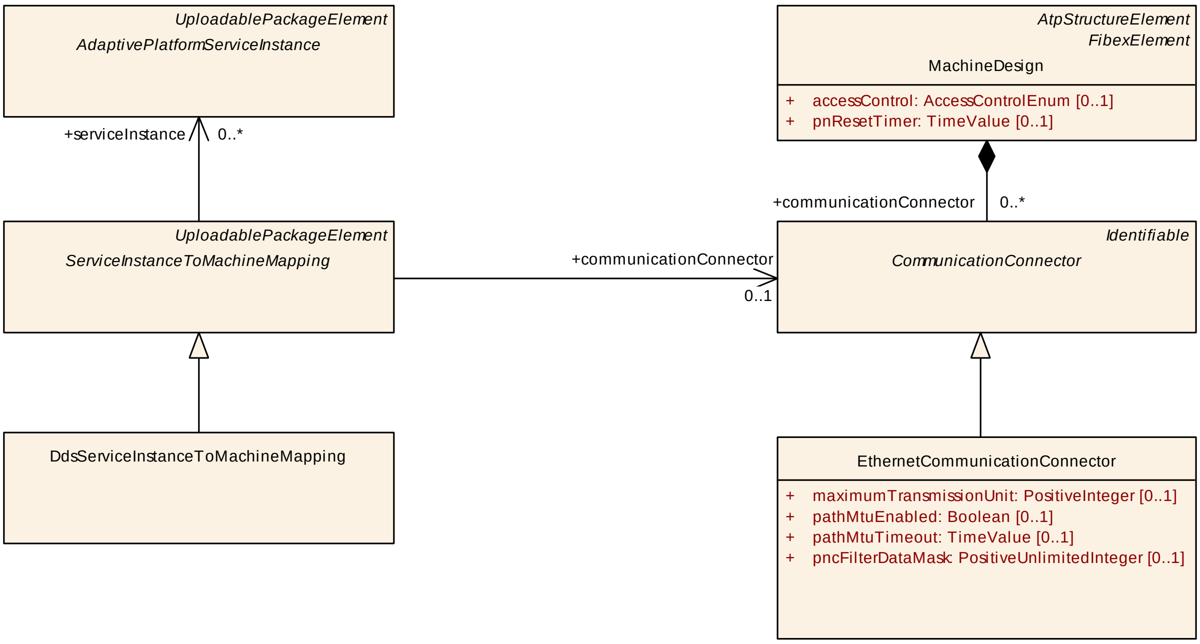


Figure 9.21: Dds Service Instance to Machine mapping

[TPS_MANI_03533]{DRAFT} **DdsServiceInstanceToMachineMapping** ⌈ The DdsServiceInstanceToMachineMapping defines for a specific serviceInstance (either DdsProvidedServiceInstance or DdsRequiredServiceInstance) on which network the communication shall be done using the reference communicationConnector to CommunicationConnector. ⌋*(RS_MANI_00038)*

Class	**DdsServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map DdsServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadablePackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 9.53: DdsServiceInstanceToMachineMapping

9.2.3 User Defined Service Instance Deployment

[TPS_MANI_03032]{DRAFT} **Description of middleware technologies not standardized by AUTOSAR** ⌈ The elements `ProvidedUserDefinedServiceInstance` and `RequiredUserDefinedServiceInstance` can be used to describe alternative middleware technologies that are not standardized by AUTOSAR. ⌋ *(RS_MANI_00014)*

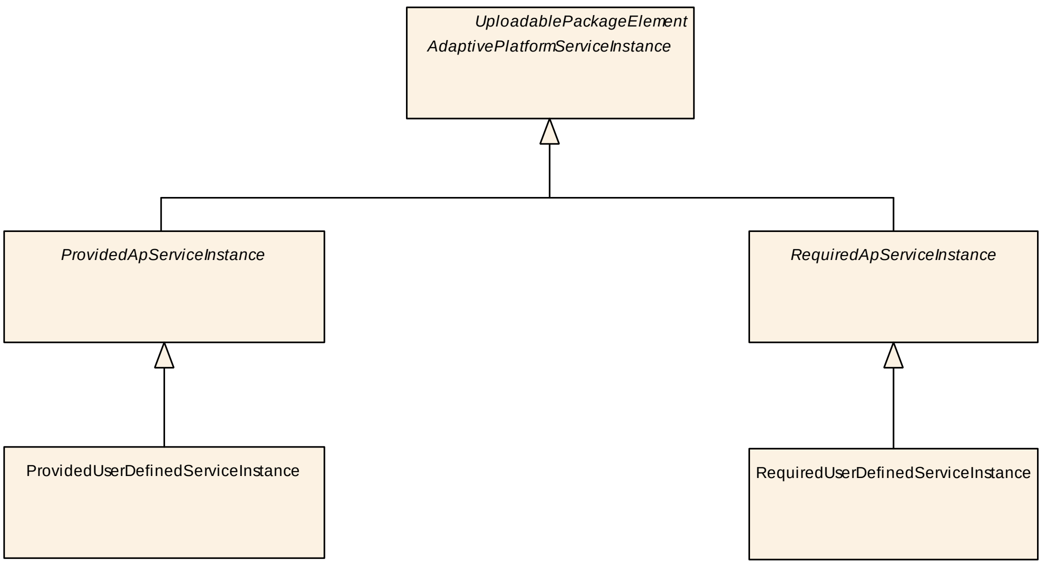


Figure 9.22: User Defined Service Instance Deployment

Class	**ProvidedUserDefinedServiceInstance**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment

▽

△

Class	ProvidedUserDefinedServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, ProvidedApServiceInstance, Referrable, Uploadable PackageElement*			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 9.54: ProvidedUserDefinedServiceInstance

Class	RequiredUserDefinedServiceInstance			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstances			
Base	*ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, RequiredApServiceInstance, Uploadable PackageElement*			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 9.55: RequiredUserDefinedServiceInstance

Please note that both elements `ProvidedUserDefinedServiceInstance` and `RequiredUserDefinedServiceInstance` are `Identifiable` and therefore are able to describe special data (`sdg`) which is not represented by the standard model.

9.3 EndToEndProtection

AUTOSAR supports the protection of `event`s and Field `notifier`s with E2E Profiles that are defined in the E2E Communication Protection Library [25].

[TPS_MANI_03127]{DRAFT} **Usage of `End2EndEventProtectionProps`** ⌈ The `End2EndEventProtectionProps` element is used to define `event` specific E2E configuration settings in the context of an `AdaptivePlatformServiceInstance`. ⌋ *(RS_MANI_00028)*

Please note that the E2E protection of a field `notifier` is possible with the `End2EndEventProtectionProps.event` reference since each specific `ServiceFieldDeployment` element aggregates a `ServiceEventDeployment` in the role `notifier`. If such an aggregated `ServiceEventDeployment` is referenced with the `End2EndEventProtectionProps.event` reference the E2E protection settings are valid for the `notifier` that is embedded by the `ServiceFieldDeployment`.

Since the End2EndEventProtectionProps element is aggregated by the abstract AdaptivePlatformServiceInstance it can be used to describe the End-to-End protection on specific derived classes like ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance that fit the underlying middleware. With this approach it is possible to define different End-to-End protection settings for different used transport layer mechanisms in case of Multi-Binding.

[TPS_MANI_03129]{DRAFT} **E2E profile** ⌈ The E2E profile is defined by E2EProfileConfiguration.profileName. ⌋*(RS_MANI_00028)*

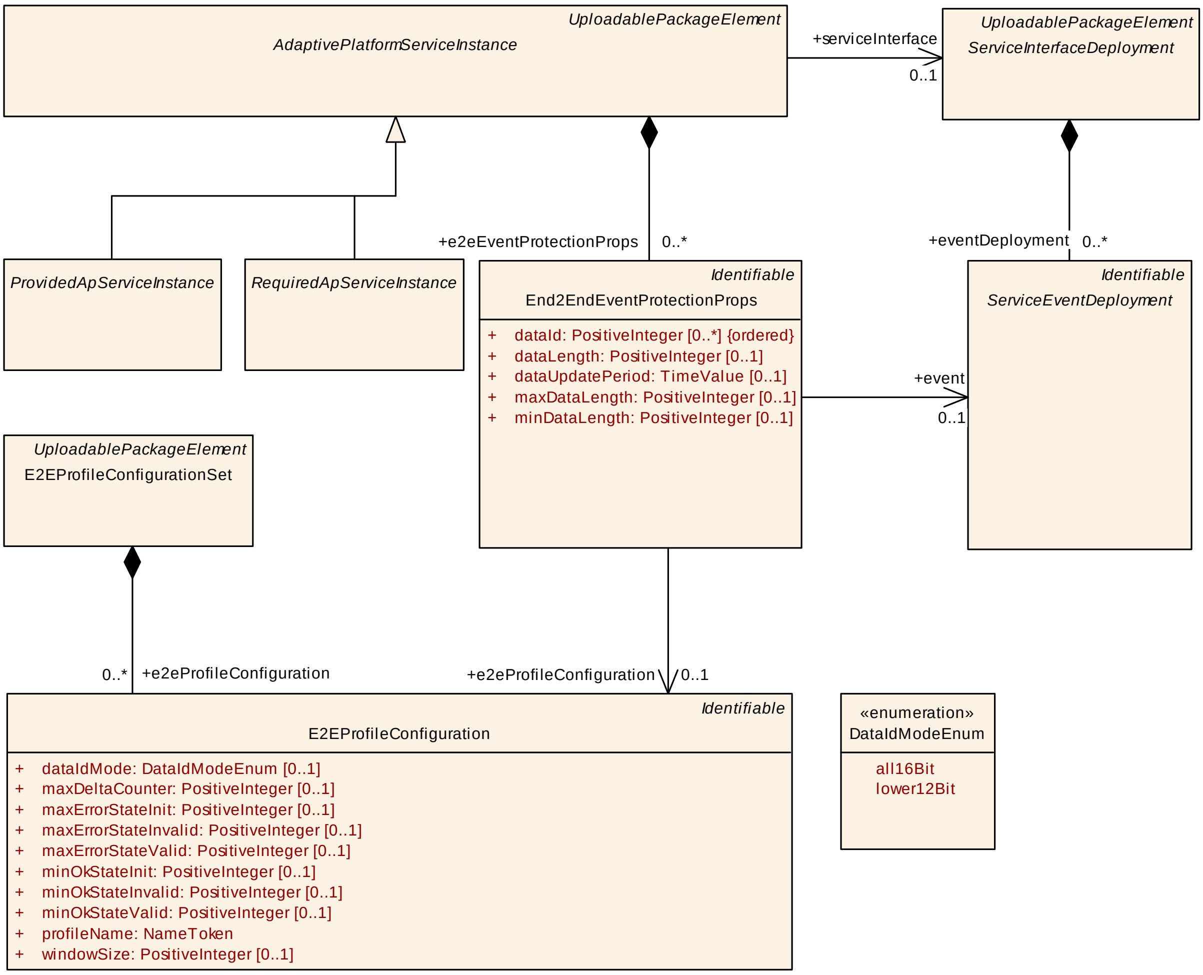


Figure 9.23: E2E EventProtection

[TPS_MANI_03130]{DRAFT} **Standardized E2EProfileConfiguration.profileName values** ⌈ The E2EProfileConfiguration.profileName that is referenced by an End2EndEventProtectionProps can have the following values that are standardized by AUTOSAR: PROFILE_04, PROFILE_05, PROFILE_06, PROFILE_07, PROFILE_11, PROFILE_22. ⌋*(RS_MANI_00028)*

[TPS_MANI_03131]{DRAFT} **Non-Standardized E2EProfileConfiguration.profileName values** ⌈ The values for the profileName of E2EProfileConfiguration mentioned in [TPS_MANI_03130] are standardized and reserved for being used in the way the AUTOSAR standard foresees.

PROFILE_01 and PROFILE_02 are also reserved by AUTOSAR but excluded for usage in Adaptive AUTOSAR. In addition, it is positively possible to use other than the standardized values for the profileName. ⌋(RS_MANI_00028)

[TPS_MANI_03128]{DRAFT} **Usage of same dataId in case of Multi-Binding** ⌈ In case of Multi-Binding, i.e. if different AdaptivePlatformServiceInstances exist that are mapped by ServiceInstanceToPortPrototypeMapping to the same PortPrototype, the different AdaptivePlatformServiceInstances may contain the same dataId for the same event. ⌋(RS_MANI_00028)

In other words if a PortPrototype contains two transport layer bindings, e.g. a ProvidedSomeipServiceInstance and a ProvidedUserDefinedServiceInstance representing an IPC communication then an event is allowed to be protected with the same dataId in both AdaptivePlatformServiceInstances.

Class	**End2EndEventProtectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This element allows to protect an event or a field notifier with an E2E profile. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataId (ordered)	PositiveInteger	*	attr	This represents a unique numerical identifier for the referenced event or field notifier that is included in the CRC calculation. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEndProtection.
dataLength	PositiveInteger	0..1	attr	Length of payload including E2E header in bits.
dataUpdatePeriod	TimeValue	0..1	attr	This attribute describes the period in which the applications are assumed to process E2E-protected messages. The middleware does not use this attribute at all.
e2eProfileConfiguration	E2EProfileConfiguration	0..1	ref	Reference to E2E profile configuration settings that are valid to protect the referenced event or field notifier. **Tags:** atp.Status=draft
event	ServiceEventDeployment	0..1	ref	Reference to an event that is protected by the E2E profile. **Tags:** atp.Status=draft
maxDataLength	PositiveInteger	0..1	attr	Maximum length of payload including E2E header in bits.
minDataLength	PositiveInteger	0..1	attr	Minimum length of payload including E2E header in bits.

Table 9.56: End2EndEventProtectionProps

Class	E2EProfileConfigurationSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This meta-class represents the ability to aggregate a collection of E2EProfileConfigurations. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=E2EProfileConfigurationSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
e2eProfileConfiguration	E2EProfileConfiguration	*	aggr	This represents the collection of E2EProfileConfigurations aggregated at the E2EProfileConfigurationSet. **Tags:** atp.Status=draft

Table 9.57: E2EProfileConfigurationSet

Class	E2EProfileConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This element holds E2E profile specific configuration settings. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataIdMode	DataIdModeEnum	0..1	attr	This attribute describes the inclusion mode that is used to include the implicit two-byte Data ID in the one-byte CRC.
maxDeltaCounter	PositiveInteger	0..1	attr	Maximum allowed difference between two counter values of two consecutively received valid messages. For example, if the receiver gets data with counter 1 and MaxDeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
maxErrorStateInit	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_INIT.
maxErrorStateInvalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_INVALID.
maxErrorStateValid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_VALID.
minOkStateInit	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INIT.
minOkStateInvalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID.
minOkStateValid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID.
profileName	NameToken	1	attr	Definition of the E2E profile.
windowSize	PositiveInteger	0..1	attr	Size of the monitoring window for the E2E state machine.

Table 9.58: E2EProfileConfiguration

Enumeration	**DataIdModeEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer
Note	Supported inclusion modes to include the implicit two-byte Data ID in the one-byte CRC.
Literal	***Description***
all16Bit	Two bytes are included in the CRC (double ID configuration). **Tags:** atp.EnumerationValue=0
lower12Bit	The low byte is included in the implicit CRC calculation, the low nibble of the high byte is transmitted along with the data (i.e. it is explicitly included), the high nibble of the high byte is not used. This is applicable for the IDs up to 12 bits. **Tags:** atp.EnumerationValue=2

Table 9.59: DataIdModeEnum

Please note that the configuration of the E2E state machines with the configuration attributes available in E2EProfileConfiguration is restricted by [constr_3176], [constr_3177], [constr_3178], [constr_3179], [constr_3180], [constr_3181] defined in the System Template [16].

It is possible to overwrite the E2E state machine configuration settings that are defined in End2EndEventProtectionProps (e2eProfileConfiguration) at the RPortPrototype of a SwComponentType with settings available in the ReceiverComSpec as described in [TPS_MANI_03132]. With this approach it is possible to define individual E2E settings for different receivers of the event or field notifier.

[constr_3493]{DRAFT} **Applicable attributes for standardized E2E Profiles** ⌈ Table 9.60 defines the applicable attributes for the standardized E2E Profiles of AUTOSAR. ⌋*()*

E2E Attributes	**Root Element**		**Attribute Existence per Profile**					
	End2EndEventProtectionProps	E2EProfileConfiguration	**PROFILE_04**	**PROFILE_05**	**PROFILE_06**	**PROFILE_07**	**PROFILE_11**	**PROFILE_22**
dataId	x		1	1	1	1	1	n
dataLength	x			x			x	x
minDataLength	x		x		x	x		
maxDataLength	x		x		x	x		
dataUpdatePeriod	x		x	x	x	x	x	x
dataIdMode		x					x	

▽

△

E2E Attributes	Root Element		Attribute Existence per Profile					
	End2EndEventProtectionProps	E2EProfileConfiguration	PROFILE_04	PROFILE_05	PROFILE_06	PROFILE_07	PROFILE_11	PROFILE_22
maxDeltaCounter		x	x	x	x	x	x	x
maxErrorStateInit		x	x	x	x	x	x	x
maxErrorStateInvalid		x	x	x	x	x	x	x
maxErrorStateValid		x	x	x	x	x	x	x
minOkStateInit		x	x	x	x	x	x	x
minOkStateInvalid		x	x	x	x	x	x	x
minOkStateValid		x	x	x	x	x	x	x
windowSize		x	x	x	x	x	x	x

Table 9.60: Allowed Attributes for standardized E2E Profiles

In PROFILE_22 the dataId is defined as a list of 16 dataId values, where a different value is transmitted depending on the counter value.

Please also note that the Classic Platform attributes counterOffset, crcOffset and dataIdNibbleOffset are not configurable in Adaptive Autosar and are set to fixed values by the AUTOSAR Standard.

9.4 Secure Communication

AUTOSAR supports different protocols that provide communication security over a network. To configure the secured communication of ServiceInterface elements between a ProvidedApServiceInstance and a RequiredApServiceInstance the ServiceInterfaceElementSecureComConfig meta-class is defined.

[TPS_MANI_03133]{DRAFT} **Usage of ServiceInterfaceElementSecureComConfig** ⌈ The ServiceInterfaceElementSecureComConfig element is used to define ServiceInterface element specific secure communication configuration settings in the context of an AdaptivePlatformServiceInstance. ⌋ *(RS_MANI_00036)*

The modeling allows to protect selected elements of a ServiceInterface, like particular events or methods.

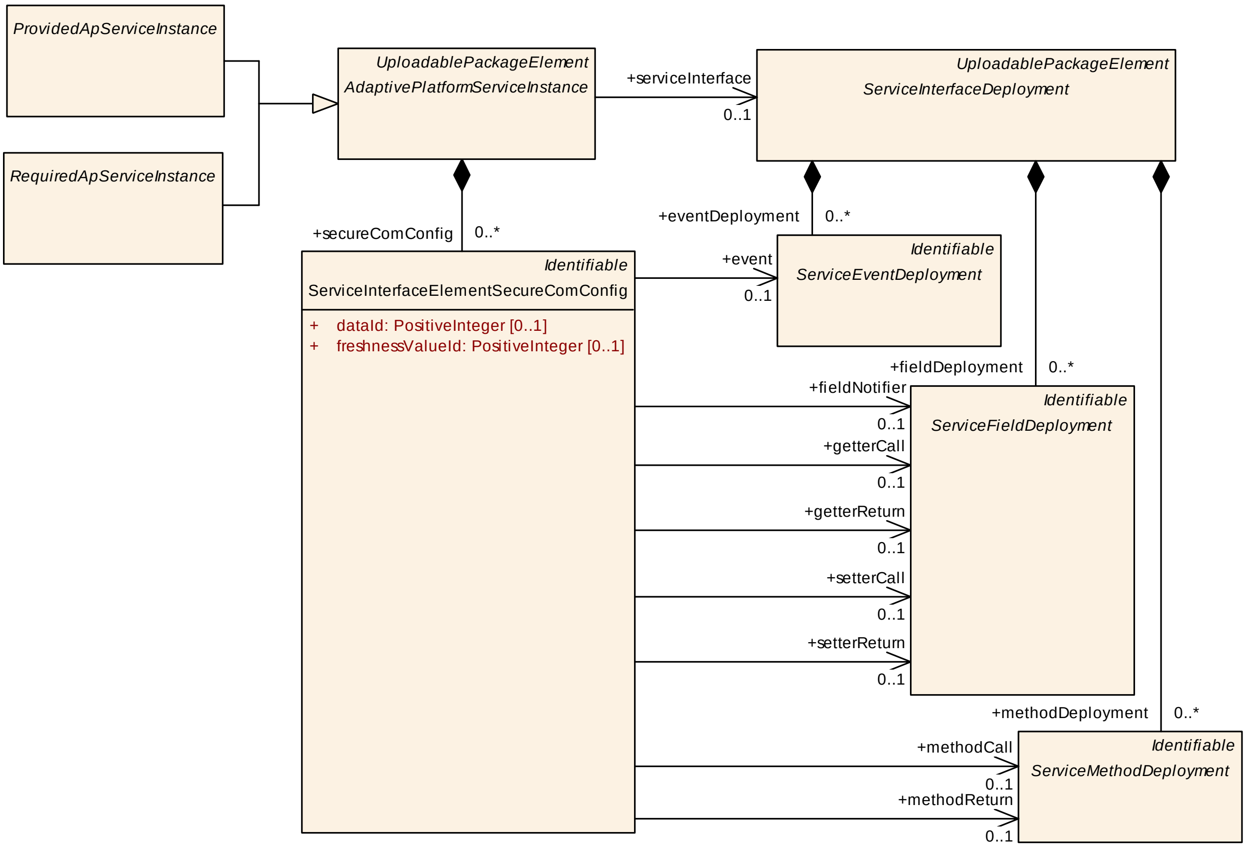


Figure 9.24: Secure Communication

Since the ServiceInterfaceElementSecureComConfig meta-class is aggregated by the abstract AdaptivePlatformServiceInstance it can be used to configure the secure communication on specific derived classes like ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance that fit the underlying middleware. With this approach it is possible to define different communication security protections for different used transport layer mechanisms in case of Multi-Binding.

Class	**ServiceInterfaceElementSecureComConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This element allows to secure the communication of the referenced ServiceInterface element. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
dataId	PositiveInteger	0..1	attr	This attribute defines a unique numerical identifier for the referenced ServiceInterface element.
event	ServiceEventDeployment	0..1	ref	Reference to an event that is protected by a security protocol. **Tags:** atp.Status=draft

▽

△

Class	**ServiceInterfaceElementSecureComConfig**			
fieldNotifier	ServiceField Deployment	0..1	ref	Reference to a field notifier that is protected by a security protocol. **Tags:** atp.Status=draft
freshnessValue Id	PositiveInteger	0..1	attr	This attribute defines the Id of the Freshness Value.
getterCall	ServiceField Deployment	0..1	ref	Reference to a field getter call message that is protected by a security protocol. **Tags:** atp.Status=draft
getterReturn	ServiceField Deployment	0..1	ref	Reference to a field getter return message that is protected by a security protocol. **Tags:** atp.Status=draft
methodCall	ServiceMethod Deployment	0..1	ref	Reference to a method call message that is protected by a security protocol. **Tags:** atp.Status=draft
methodReturn	ServiceMethod Deployment	0..1	ref	Reference to a method return message that is protected by a security protocol. **Tags:** atp.Status=draft
setterCall	ServiceField Deployment	0..1	ref	Reference to a field setter call message that is protected by a security protocol. **Tags:** atp.Status=draft
setterReturn	ServiceField Deployment	0..1	ref	Reference to a field setter return message that is protected by a security protocol. **Tags:** atp.Status=draft

Table 9.61: ServiceInterfaceElementSecureComConfig

[constr_3391]{DRAFT} **ServiceInterfaceElementSecureComConfig references to ServiceInterfaceDeployment elements** ⌈ ServiceInterfaceElementSecureComConfig element shall be defined for exactly one ServiceInterface element and shall therefore contain only one single reference to an element defined in the scope of a ServiceInterfaceDeployment. ⌋*()*

The attributes in the ServiceInterfaceElementSecureComConfig meta-class are defining security configuration settings that are specific for the referenced ServiceInterface element in the context of an AdaptivePlatformServiceInstance. The used security protocol is defined in the ServiceInstanceToMachineMapping.

[TPS_MANI_03199]{DRAFT} **Endpoint protection by SecureComProps** ⌈ The ServiceInstanceToMachineMapping allows to assign security protocol configuration settings that are defined in the referenced SecureComProps meta-class to protect endpoints that are defined by the Transport Protocol, Port and IP Address on which one or several AdaptivePlatformServiceInstances are provided or consumed. ⌋*(RS_MANI_00036)*

[TPS_MANI_03200]{DRAFT} **SecureComProps for udp, tcp and multicast communication** ⌈ The ServiceInstanceToMachineMapping allows to assign security protocol configuration settings for:

- udp communication if the ServiceInstanceToMachineMapping refers the SecureComProps in the role secureComPropsForUdp

- tcp communication in case the ServiceInstanceToMachineMapping refers the SecureComProps in the role secureComPropsForTcp
- multicast communication in case the ServiceInstanceToMachineMapping refers the SecOcSecureComProps in the role secOcComPropsForMulticast

⌋(RS_MANI_00036)

Please note that protection of IP multicast traffic is only supported by SecOC and therefore the ServiceInstanceToMachineMapping refers directly the SecOcSecureComProps in the secOcComPropsForMulticast role.

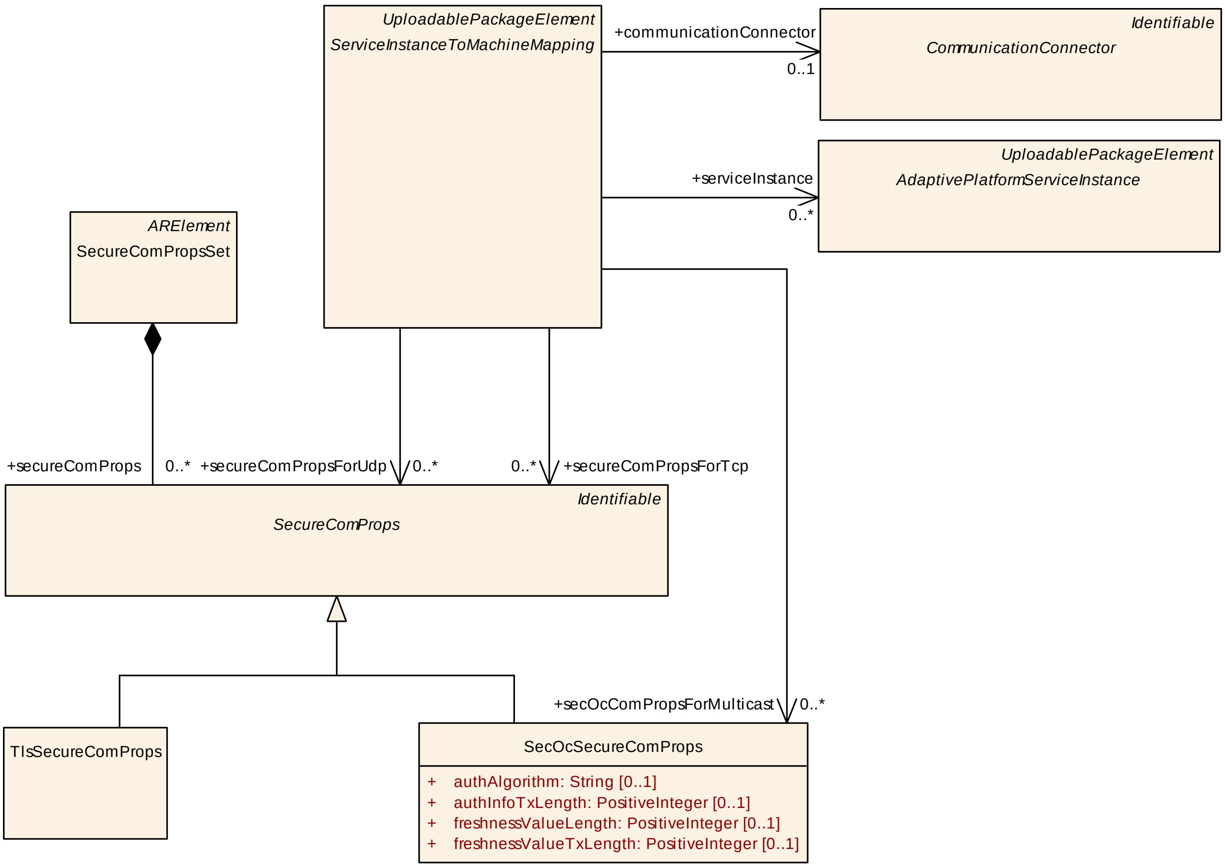


Figure 9.25: Security protocol configuration

Class	**SecureComPropsSet**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	This meta-class represents the ability to aggregate a collection of SecureComProps.. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=SecureComPropsSets
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	**SecureComPropsSet**			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
secureCom Props	SecureComProps	*	aggr	This represents the collection of SecureComProps aggregated at the SecureComPropsSet. **Tags:** atp.Status=draft

Table 9.62: SecureComPropsSet

Class	***SecureComProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This meta-class defines a communication security protocol and its configuration settings. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SecOcSecureComProps, TlsSecureComProps			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 9.63: SecureComProps

9.4.1 Secure Communication over TLS

The configuration of the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols is supported with the `TlsSecureComProps` meta-class, which is a specialization of `SecureComProps`.

It is a common use case that only one end of a TLS-based connection is actually modeled in an AUTOSAR model. It is therefore important that the modeling does not rely on or imply knowledge about both ends of such a TLS-based connection.

An AUTOSAR model that only describes one end of the communication is positively required to work, independently of the availability of a formal modeling of the other end.

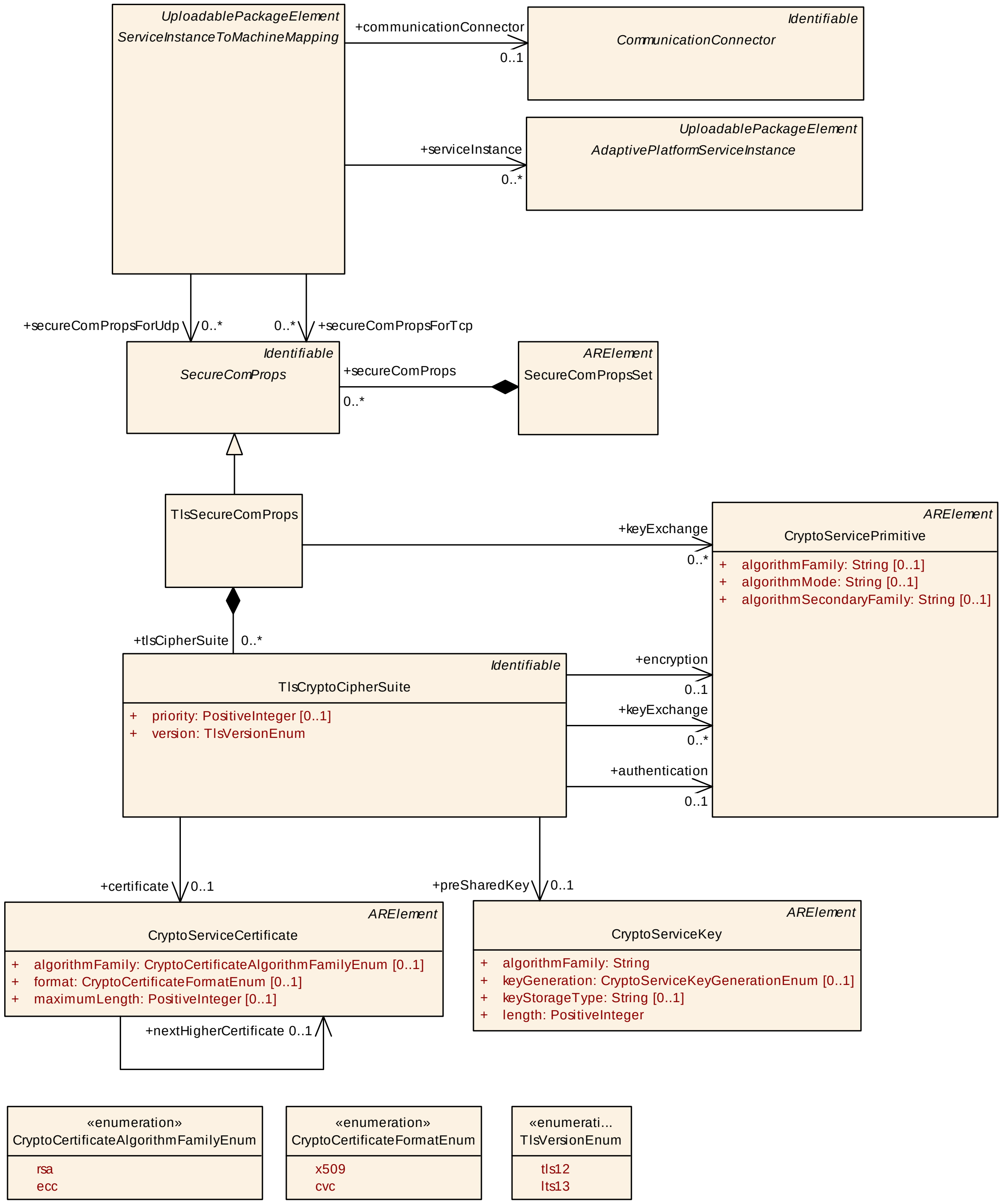


Figure 9.26: Secure Communication over TLS

Class	TlsSecureComProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	Configuration of the Transport Layer Security protocol (TLS). **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SecureComProps*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared (i.e. applicable for each of the aggreated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase. **Tags:** atp.Status=draft
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	Collection of supported cipher suites that are used to negotiate the security settings for a network connection defined by the ServiceInstanceToMachineMapping. **Tags:** atp.Status=draft

Table 9.64: TlsSecureComProps

Enumeration	CryptoServiceKeyGenerationEnum
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enumeration shall be taken to express the handling of a crypto key in terms of whether it is obtained from e.g. a diagnostic tester or whether it is created by derivation from a master key.
Literal	***Description***
keyDerivation	This means that the crypto key is created by derivation from a master key. **Tags:** atp.EnumerationValue=0
keyStorage	This means that the crypto key is obtained from an external entity, e.g. a diagnostic tester. **Tags:** atp.EnumerationValue=1

Table 9.65: CryptoServiceKeyGenerationEnum

TLS is composed of the TLS Record Protocol and the TLS Handshake Protocol. The Record Protocol provides connection security and encrypts and authenticate packets. The record layer functions can be called at any time after the handshake process is finished, when there is need to receive or send data.

The Handshake Protocol allows the server and client to authenticate each other and to negotiate encryption algorithms and cryptographic keys before any data is exchanged.

In order to establish a cryptographically secure data channel, the communication partners in form of `ServiceInstanceToMachineMapping`s must agree on ciphersuites and on keys that will be used to encrypt the data.

The client sends a list of supported ciphersuites to the server. The server decides on a ciphersuite from the list provided by the client, and continues with the handshake. Please note that the server and client roles cannot be swapped while the connection exists, i.e. a *server* remains the *server* for the full amount of time the connection exists.

[TPS_MANI_03213]{DRAFT} **Semantics of meta-class `TlsSecureComProps`** ⌈ As a sub-class of `SecureComProps`, meta-class `TlsSecureComProps` has the ability to collect the TLS-related configuration aspects from either the perspective of the client or the server.

In the case of TLS, the collection boils down to the aggregation of meta-class TlsCryptoCipherSuite in the role tlsCipherSuite plus the ability (by means of the role keyExchange) to define handshake properties that are shared for each of the aggregated tlsCipherSuite. ⌋*(RS_MANI_00036)*

[constr_5047]{DRAFT} **Supported values of ServiceInstanceToMachineMapping.category** ⌈ The only supported values of attribute TlsSecureComProps.category are:

- **TLS_SERVER**: the TlsSecureComProps assumes the role of the *server* in the TLS connection.
- **TLS_CLIENT**: the TlsSecureComProps assumes the role of the *client* in the TLS connection.

⌋*()*

[TPS_MANI_03134]{DRAFT} **Configuration of supported TLS ciphersuites** ⌈ The creation of a TLS connection requires the usage of a suite of cryptographic operations in specific roles, also known as a *cipher suite*.

Meta-class TlsCryptoCipherSuite represents a given cipher suite for a TLS connection. TlsCryptoCipherSuite references meta-class CryptoServicePrimitive in three dedicated roles that represent the steps of the creation of a TLS connection.

More specifically, the cryptographic operations for setting up a TLS connection involve the following steps:

- **Key exchange**: these CryptoServicePrimitives may be used for the handshake phase of the TLS connection. Different alternatives exist for executing this phase and therefore the multiplicity of this reference is 0..*.
- **Authentication** of communication partners during the operational phase of the TLS connection. For this purpose a single CryptoServicePrimitive is used on each end of the communication.
- **Encryption** of content exchanged between the communication partners that have established the TLS connection. For this purpose a single CryptoServicePrimitive is used on each end of the communication.

⌋*(RS_MANI_00036)*

Class	**TlsCryptoCipherSuite**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class represents a cipher suite for describing cryptographic operations in the context of establishing a connection of ApplicationEndpoints that is protected by TLS. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
authentication	CryptoServicePrimitive	0..1	ref	This reference identifies the crypto service primitive for the generation and verification of MACs. **Tags:** atp.Status=draft
certificate	CryptoService Certificate	0..1	ref	This reference identifies the applicable certificate. **Tags:** atp.Status=draft
encryption	CryptoServicePrimitive	0..1	ref	This reference identifies the crypto service primitive for the execution of encryption. **Tags:** atp.Status=draft
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the individual (i.e. per cipher suite) crypto service primitive for the execution of key exchange during the handshake phase. **Tags:** atp.Status=draft
preSharedKey	CryptoServiceKey	0..1	ref	This reference identifies the applicable cryptograhic key if the handshake is based on the existence of a pre-shared key (PSK) **Tags:** atp.Status=draft
priority	PositiveInteger	0..1	attr	This attribute identifies the priority of the cipher suite. Range: 1..65535. Lower values represent higher priorities.
version	TlsVersionEnum	1	attr	This attribute supports the definition of the applicable version of TLS.

Table 9.66: TlsCryptoCipherSuite

Class	**CryptoServicePrimitive**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto primitive. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.recommendedPackage=CryptoPrimitives			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
algorithmFamily	String	0..1	attr	This attribute represents a description of the family (e.g. AES) of crypto algorithm implemented by the crypto primitive.
algorithmMode	String	0..1	attr	This attribute represents a description of the mode of the crypto algorithm implemented by the crypto primitive.

▽

△

Class	**CryptoServicePrimitive**			
algorithm Secondary Family	String	0..1	attr	This attribute represents a further description of the secondary family of crypto algorithm implemented by the crypto primitive. The secondary family is needed for the specification of the hash algorithm for a signature check, e.g. using RSA.

Table 9.67: CryptoServicePrimitive

Class	**CryptoServiceKey**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto key **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.recommendedPackage=CryptoDevelopmentKeys			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
algorithmFamily	String	1	attr	This attribute represent the description of the family of the applicable crypto algorithm.
development Value	ValueSpecification	0..1	aggr	This aggregation represents the ability to assign a specific value to the crypto key as part of the system description. This value can then be taken for the development of the respective ECU.
keyGeneration	CryptoServiceKey GenerationEnum	0..1	attr	This attribute describes how a the specific cryptographic key is created.
keyStorageType	String	0..1	attr	This attribute describes where the enclosing cryptographic key shall be stored. AUTOSAR reserves specific values for this attributes but it is possible to insert custom values as well.
length	PositiveInteger	1	attr	This attribute describes the length of the cryptographic key.

Table 9.68: CryptoServiceKey

[TPS_MANI_03214]{DRAFT} **Existence of `TlsCryptoCipherSuite.keyExchange` vs. `TlsSecureComProps.keyExchange`** ⌈ The role `TlsSecureComProps.keyExchange` has been introduced as an optimization.

It is assumed that the references for key exchange look pretty similar if not identical for many concrete `TlsCryptoCipherSuite`s.

Adding these references in an identical form to a bunch of `TlsCryptoCipherSuite`s does not really make sense. Therefore, `TlsSecureComProps` allows to define these references as well with the intention to make them valid for all `TlsSecureComProps.tlsCipherSuite`s.

A mixture of references in the role `TlsCryptoCipherSuite.keyExchange` and `TlsSecureComProps.keyExchange` is supported. ⌋*(RS_MANI_00036)*

[TPS_MANI_03215]{DRAFT} **Semantics of `CryptoServiceCertificate`** ⌈ Meta-class `CryptoServiceCertificate` represents a cryptographic certificate needed for the creation of a TLS connection between *server* and *client*. ⌋*()*

Class	CryptoServiceCertificate			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class represents the ability to model a cryptographic certificate. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=CryptoServiceCertificates			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
algorithmFamily	CryptoCertificate AlgorithmFamilyEnum	0..1	attr	This attribute represents a description of the family of crypto algorithm used to generate public key and signature of the cryptographic certificate.
format	CryptoCertificateFormat Enum	0..1	attr	This attribute can be used to provide information about the format used to create the certificate
maximum Length	PositiveInteger	0..1	attr	This attribute represents the ability to define the maximum length of the certificate.
nextHigher Certificate	CryptoService Certificate	0..1	ref	The reference identifies the next higher certificate in the certificate chain. **Tags:** atp.Status=draft

Table 9.69: CryptoServiceCertificate

Enumeration	CryptoCertificateAlgorithmFamilyEnum
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This meta-class defies possible cryptographic algorithm families used to create public keys and signatures within the certificate. **Tags:** atp.Status=draft
Literal	**Description**
ecc	The cryptographic operations in the certificate are executed using elliptic curves (ecc) **Tags:** atp.EnumerationValue=2
rsa	The cryptographic operations in the certificate are executed using the RSA approach. **Tags:** atp.EnumerationValue=1

Table 9.70: CryptoCertificateAlgorithmFamilyEnum

Enumeration	CryptoCertificateFormatEnum
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This meta-class defines possible formats of cryptographic certificates. **Tags:** atp.Status=draft
Literal	**Description**
cvc	The certificate has been created in Card Verifiable Certificate (CVC) format **Tags:** atp.EnumerationValue=2
x509	The certificate is created in X.509 format. **Tags:** atp.EnumerationValue=1

Table 9.71: CryptoCertificateFormatEnum

The existence of a certificate is required for the *server* role in order to be able to authenticate itself towards the *client*.

[constr_5048]{DRAFT} **Existence of `TlsCryptoCipherSuite.certificate` in the *server* role** ⌈ **Either of** the references to

- `CryptoServiceCertificate` in the role `TlsCryptoCipherSuite.certificate`
- `CryptoServiceKey` in the role `TlsCryptoCipherSuite.preSharedKey`

shall exist if the `CryptoServiceCertificate` is referenced from a `ServiceInstanceToMachineMapping` that has attribute `category` set to the value `TLS_SERVER`. ⌋*()*

The *client* may also use a certificate to authenticate itself to the *server*, but this is not mandatory. If this option is not used then other documented approaches for completing the handshake phase is foreseen for the specific case.

[TPS_MANI_03216]{DRAFT} **Existence of `TlsCryptoCipherSuite.certificate` in the *client* role** ⌈ **Either of** the references to

- `CryptoServiceCertificate` in the role `TlsCryptoCipherSuite.certificate`
- `CryptoServiceKey` in the role `TlsCryptoCipherSuite.preSharedKey`

may exist if the `CryptoServiceCertificate` is referenced from a `ServiceInstanceToMachineMapping` that has attribute `category` set to the value `TLS_CLIENT`. ⌋*(RS_MANI_00036)*

If the reference exists then the *client* intends to authenticate itself against the server during the handshake phase using a cryptographic certificate.

[TPS_MANI_03137]{DRAFT} **`ServiceInterfaceElementSecureComConfig` is not relevant in case of TLS communication** ⌈ The element `ServiceInterfaceElementSecureComConfig` is not relevant in case of TLS communication. ⌋*(RS_MANI_00036)*

[constr_3485]{DRAFT} **UDP endpoint using DTLS can only serve provided or required service instances exclusively** ⌈ A `ServiceInstanceToMachineMapping` that refers to `TlsSecureComProps` in the role `secureComPropsForUdp` is not allowed to refer to `ProvidedApServiceInstance`s and at the same time to `RequiredApServiceInstance`s in the role `serviceInstance`. In other words a UDP endpoint using DTLS can only serve provided or required service instances exclusively. ⌋*()*

[constr_3486]{DRAFT} **TCP endpoint using TLS can only serve provided or required service instances exclusively.** ⌈ A `ServiceInstanceToMachineMapping` that refers to `TlsSecureComProps` in the role `secureComPropsForTcp` is not allowed to refer to `ProvidedApServiceInstance`s and at the same time to `RequiredApServiceInstance`s in the role `serviceInstance`. In other words a TCP endpoint using TLS can only serve provided or required service instances exclusively. ⌋*()*

The reason for [constr_3485] and [constr_3486] is that the (D)TLS client needs to establish the (D)TLS connection and a TCP/UDP endpoint that is described by the ServiceInstanceToMachineMapping can only take one role: (D)TLS client or (D)TLS server. If a ServiceInstanceToMachineMapping would act as (D)TLS client and would refer to a ProvidedApServiceInstance then this (D)TLS client would need to establish the (D)TLS connection. But in this case the (D)TLS client would not know to which remote service client a connection needs to be established since different RequiredApServiceInstances may directly call methods of the ProvidedApServiceInstance without any registration.

The same issue exists if the ServiceInstanceToMachineMapping acts as (D)TLS server and refers to RequiredApServiceInstances. The (D)TLS client needs to establish the (D)TLS connection before any messages are exchanged. But the remote service provider has no knowledge that this service consumer wants to call methods over a (D)TLS connection.

9.4.2 Secure Communication over SecOC

AUTOSAR Secure Onboard Communication (SecOC) supports symmetric and asymmetric authentication approaches. To configure the SecOC secure protection of a message by a MAC or Signature the ServiceInterfaceElementSecureComConfig needs to be defined. This element contains the configuration settings for the individual ServiceInterface elements. In addition the ServiceInstanceToMachineMapping needs to point to SecOcSecureComProps to configure the endpoint protection that is defined by the Transport Protocol, Port and IP Address.

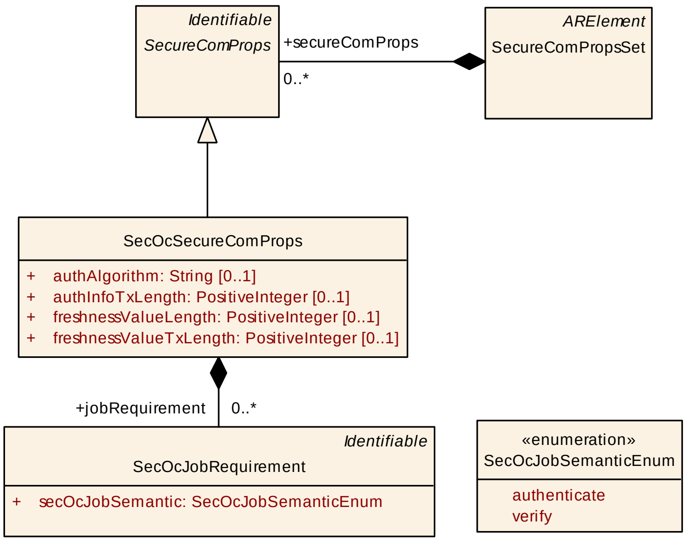


Figure 9.27: Secure Communication over SecOC

[constr_3392]{DRAFT} **ServiceInterfaceElementSecureComConfig.dataId** and **ServiceInterfaceElementSecureComConfig.freshnessValueId** are

mandatory in case of SecOC communication ⌈ The attributes `ServiceInterfaceElementSecureComConfig.dataId` and `ServiceInterfaceElementSecureComConfig.freshnessValueId` are mandatory in case of SecOC communication. ⌋*()*

[TPS_MANI_03138]{DRAFT} **SecOC Security Profile** ⌈ The SecOC security profile is defined by `SecOcSecureComProps.category`. ⌋*(RS_MANI_00036)*

Class	**SecOcSecureComProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	Configuration of AUTOSAR SecOC. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable, SecureComProps*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
authAlgorithm	String	0..1	attr	This attribute defines the authentication algorithm used for MAC generation and verification.
authInfoTxLength	PositiveInteger	0..1	attr	This attribute defines the length in bits of the authentication code to be included in the payload of the authenticated Message.
freshnessValueLength	PositiveInteger	0..1	attr	This attribute defines the complete length in bits of the Freshness Value.
freshnessValueTxLength	PositiveInteger	0..1	attr	This attribute defines the length in bits of the Freshness Value to be included in the payload of the secured message. In other words this attribute defines the length of the authenticated Message.
jobRequirement	SecOcJobRequirement	*	aggr	Collection of cryptographic job requirements. **Tags:** atp.Status=draft

Table 9.72: SecOcSecureComProps

[TPS_MANI_03139]{DRAFT} **Standardized SecOC Security Profiles** ⌈ The SecOC security profile that is defined by `SecOcSecureComProps.category` can have the following values that are standardized by AUTOSAR: PROFILE_01, PROFILE_02, PROFILE_03. ⌋*(RS_MANI_00036)*

The attribute values for the predefined categories mentioned in [TPS_MANI_03139] are defined in [constr_3325] in [16].

[TPS_MANI_03140]{DRAFT} **Non-Standardized SecOC Security Profiles** ⌈ The values for the `SecOcSecureComProps.category` mentioned in [TPS_MANI_03139] are standardized and reserved for being used in the way the AUTOSAR standard foresees. In addition, it is positively possible to use other than the standardized values for the `SecOcSecureComProps.category`. ⌋*(RS_MANI_00036)*

With the `SecOcJobRequirement` the cryptographic routines can be selected that need to be supported. In case of SecOC it can be selected whether the symmetric and/or asymmetric authentication approach is needed.

Class	SecOcJobRequirement			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	Requirements for the cryptographic job that need to be executed. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
secOcJob Semantic	SecOcJobSemantic Enum	1	attr	This attribute defines the cryptographic algorithm that needs to be supported.

Table 9.73: SecOcJobRequirement

Enumeration	SecOcJobSemanticEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	List of cryptographic routines supported by SecOC. **Tags:** atp.Status=draft
Literal	***Description***
authenticate	Authentication algorithm for Authenticator generation/verification. **Tags:** atp.EnumerationValue=0
verify	Asymmetric cryptographic algorithm to generate/verify a signature **Tags:** atp.EnumerationValue=1

Table 9.74: SecOcJobSemanticEnum

9.5 Log and Trace

The Log and Trace functionality in AUTOSAR supports the monitoring of applications and provides means to forward logging information onto the communication bus, the console, or to the file system. The logging information is put into a standardized delivery and presentation format that is described in more detail in the Log and Trace Protocol specification [26]. The format contains meta-data that identifies for example the application that produces the logging information.

This chapter describes settings that are available in the Execution Manifest and Service Instance Manifest to configure the logging framework. Some of these settings will be put as meta-data into the Log and Trace messages.

[TPS_MANI_03160]{DRAFT} **Log and Trace configuration options in the Execution Manifest** ⌈

- The `logTraceProcessId` in the `Process` identifies the application instance and is put as `ApplicationId` into the log and trace message.
- The `logTraceProcessDesc` in the `Process` is an optional setting that allows to describe the `logTraceProcessId` as descriptive text.
- `logTraceLogMode` in the `Process` defines the destination to which the log messages will be forwarded.

- logTraceDefaultLogLevel in the Process defines the initial log reporting level for the application instance. The log level defines the severity grade of the Log Message.
- logTraceFilePath in the Process This attribute defines the destination file to which the logging information is passed.

⌋*(RS_MANI_00037)*

[constr_3426]{DRAFT} **The logTraceFilePath is mandatory in case that logTraceLogMode is set to file** ⌈ In case that the logTraceLogMode is set to file the logTraceFilePath shall be set to a value. ⌋*()*

[constr_3427]{DRAFT} **The logTraceFilePath is only relevant if logTraceLogMode is set to file** ⌈ The logTraceFilePath shall only be used if logTraceLogMode is set to file. ⌋*()*

[TPS_MANI_03161]{DRAFT} **Log and Trace configuration options in the Service Instance Manifest** ⌈

- The enablesLogTrace flag in the ServiceInstanceToPortPrototypeMapping enables or disables the tracing of the communication over the referenced PortPrototype of the referenced Process.
- The logTracePortId identifies the communication of the referenced Process and is put as ContextId into the log and trace message.

⌋*(RS_MANI_00037)*

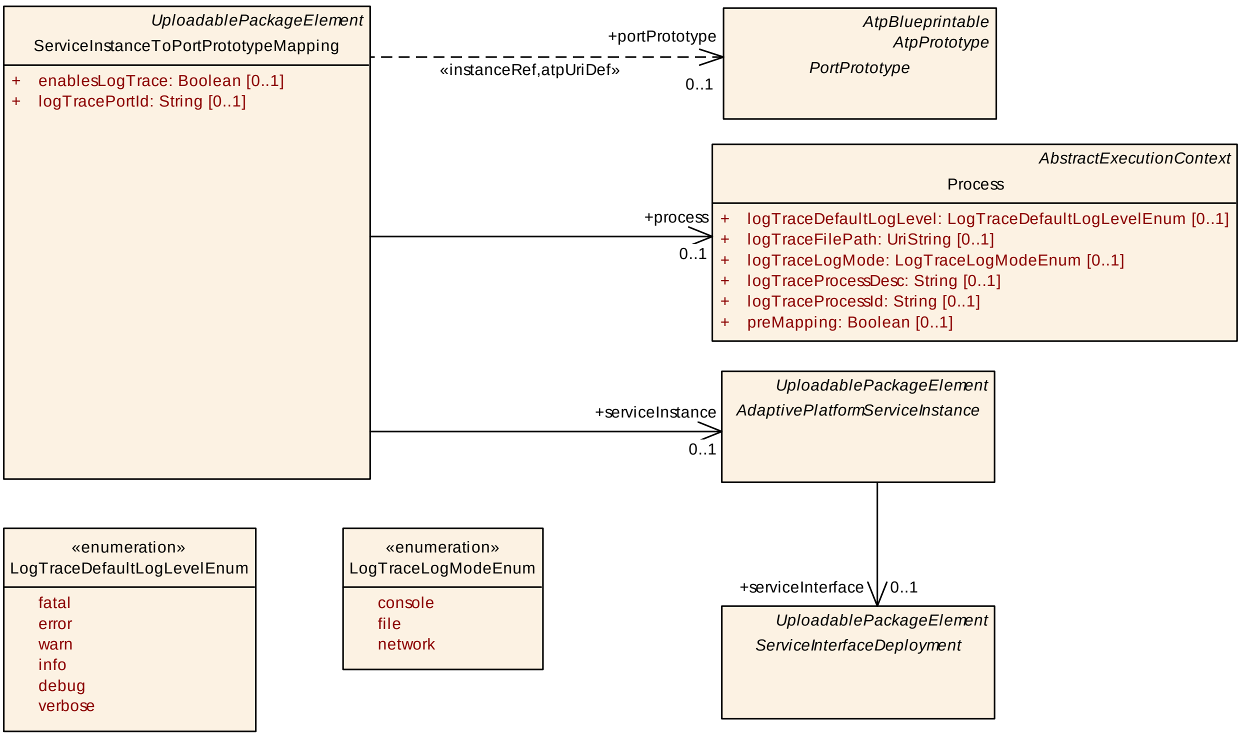


Figure 9.28: Log and Trace configuration in Application- and ServiceInstanceManifest

The output channel on Ethernet for the Log and Trace messages is configured in the Machine Manifest. The LogAndTraceInstantiation aggregates a NetworkConfiguration in the role networkConfiguration that allows to specify over which Transport Protocol (Udp or Tcp), Port and IP Address the messages are transported.

Enumeration	**LogTraceLogModeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest
Note	This enum defines the possible destinations of a log&trace message. **Tags:** atp.Status=draft
Literal	***Description***
console	Destination of log message will be the console output. **Tags:** atp.EnumerationValue=0
file	Destination of log message will be a file on the file system. **Tags:** atp.EnumerationValue=1
network	Log message will be transmitted over the communication bus. **Tags:** atp.EnumerationValue=2

Table 9.75: LogTraceLogModeEnum

Enumeration	**LogTraceDefaultLogLevelEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest
Note	This enum defines available log&trace log levels that may be used to define the severity level of a log message. **Tags:** atp.Status=draft
Literal	***Description***
debug	Detailed information for programmers **Tags:** atp.EnumerationValue=4
error	Error with impact to correct functionality **Tags:** atp.EnumerationValue=1
fatal	Fatal error **Tags:** atp.EnumerationValue=0
info	High level information **Tags:** atp.EnumerationValue=3
verbose	Verbose debug message **Tags:** atp.EnumerationValue=5
warn	Warning if correct behavior cannot be ensured **Tags:** atp.EnumerationValue=2

Table 9.76: LogTraceDefaultLogLevelEnum

10 Signal-based communication

10.1 Overview

The applications on the adaptive platform communicate with each other in a service-oriented manner. But there is also a use case where applications on the *AUTOSAR adaptive platform* need to communicate with software-components running on the *AUTOSAR classic platform*.

If the remote ECU on the *AUTOSAR classic platform* communicates via SOME/IP in a service-oriented manner and uses the SOME/IP transformer to serialize its data, then the communication with the Machine on the *AUTOSAR adaptive platform* can be established directly without any adaptations of neither the ECU nor the Machine.

If the counterpart on the *AUTOSAR classic platform* ECU communicates only using signal-based communication over, e.g., CAN or FlexRay, the translation of the signal-based content into ServiceInterfaces needs to be established.

Such a Signal-to-Service translation may happen in a Gateway that is implemented on an ECU on the *AUTOSAR classic platform*. Such a solution is out of scope of this document since it is handled using the *AUTOSAR classic platform* configuration means.

Another alternative for this translation is to happen directly on the Machine on the *AUTOSAR adaptive platform* by an Application that is running in the Process, as sketched in Figure 10.1.

This Application communicates with other applications on the *AUTOSAR adaptive platform* in the service-oriented way over ara::com; but it is also able to transmit and receive ISignals as well as communicate signal-based with remote ECUs on the *AUTOSAR classic platform*.

In order to make this possible, software that conforms to the specification of the COM stack on the *AUTOSAR classic platform* needs to be executed on the Machine on the *AUTOSAR adaptive platform*.

For the configuration of this software, the System Description based on the System Template on the *AUTOSAR classic platform* is used that contains a communication matrix description with Pdus and ISignals.

This chapter introduces a modeling that creates a bridge between the service-oriented communication based on ServiceInterfaces of the *AUTOSAR adaptive platform* and the signal-based communication involving the definition of Pdus and ISignals that are used on the *AUTOSAR classic platform*.

The Signal-to-Service mapping, together with the *AUTOSAR classic platform* System Description, allows to configure the communication between a Machine on the *AUTOSAR adaptive platform* and an ECU on the *AUTOSAR classic platform*. Please note that in a setup like the one sketched in Figure 10.1, the *AUTOSAR classic platform* System Description would also contain a Pdu or Signal Gateway configuration between the Ethernet and the CAN bus.

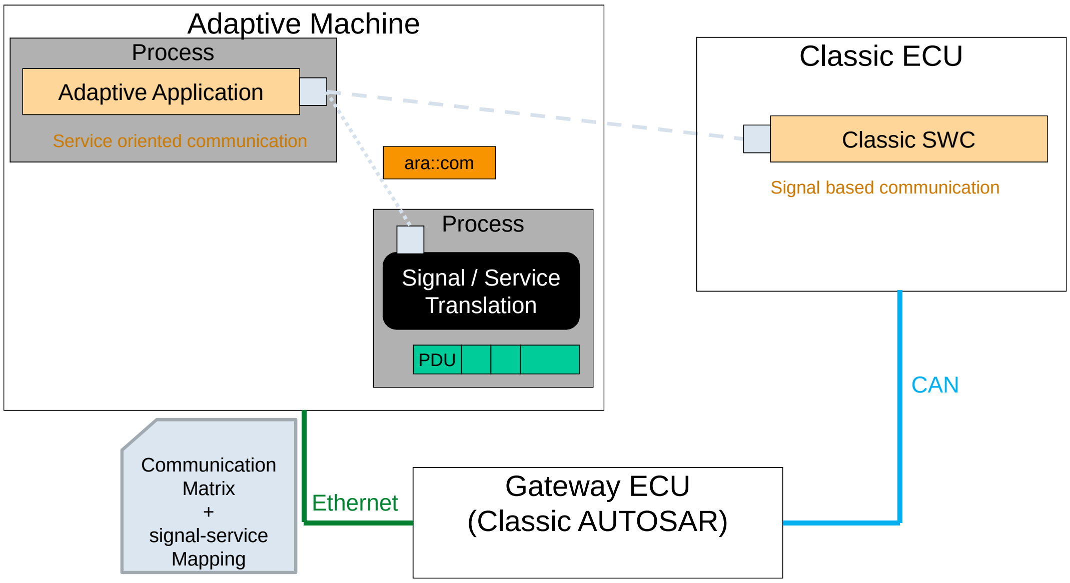


Figure 10.1: SignalToService translation in Application on Adaptive Machine

Please note that the configuration of such signal-based communication on an adaptive machine may be solved in two different ways:

1. The communication matrix definition (ARXML System Description) and the Signal-to-Service mapping is available on the target machine and is interpreted at run-time (like the manifest approach).
2. The communication matrix definition (ARXML System Description) and the Signal-to-Service mapping is built off-board and the application executable gets uploaded to the target `Machine` in response to changes in the communication matrix.

10.2 Signal-based Deployment

The `SignalBasedServiceInterfaceDeployment`, as a specialization of `ServiceInterfaceDeployment`, allows to express that the `ServiceInterface` referenced in the role `serviceInterface` will be transmitted in the signal-based way over a communication medium.

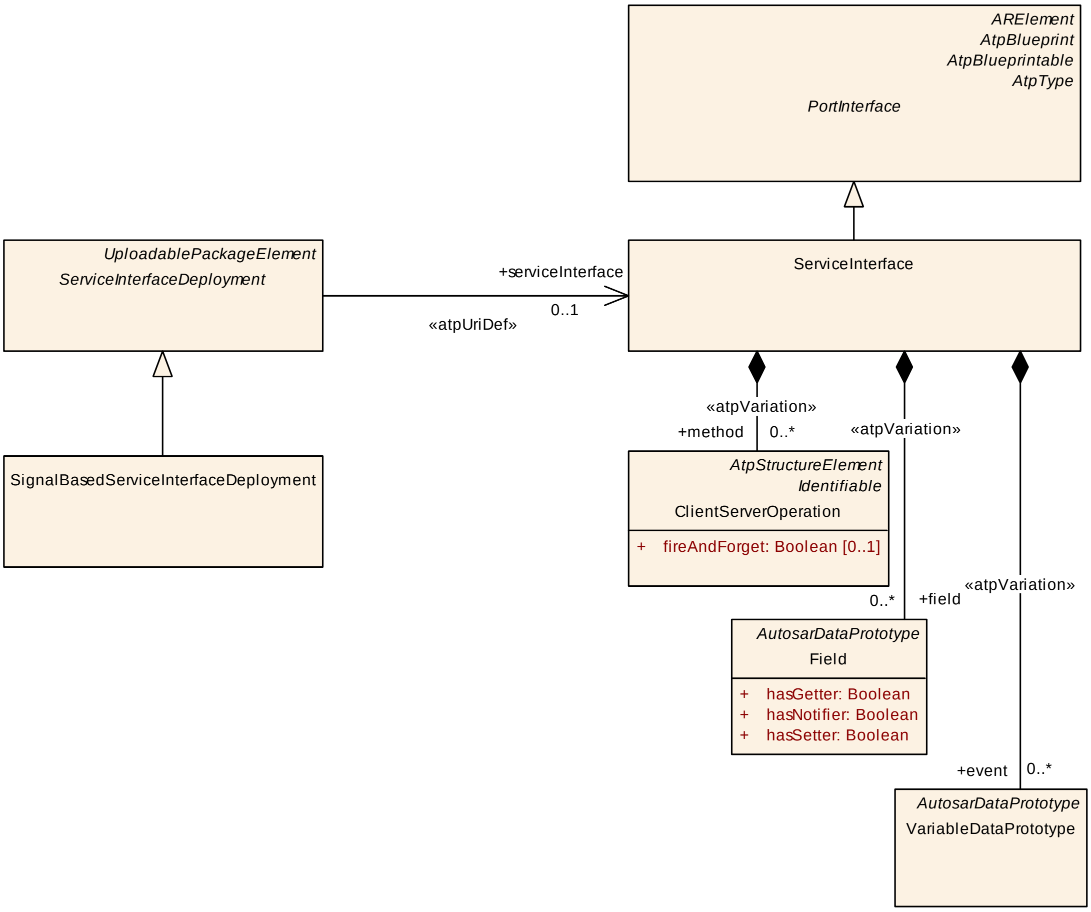


Figure 10.2: Signal-based deployment of ServiceInterface

[TPS_MANI_03120]{DRAFT} **Signal-based ServiceInterface binding** ⌈ The SignalBasedServiceInterfaceDeployment meta-class provides the ability to bind a ServiceInterface that is referenced in the role serviceInterface to a signal-based communication protocol like CAN or FlexRay. ⌋*(RS_MANI_00029)*

Please note that in contrast to other ServiceInterfaceDeployments that are described in section 9.1, the communication is not described with AdaptivePlatformServiceInstance elements but with a Signal-to-Service Mapping and a classic platform System Description.

Class	**SignalBasedServiceInterfaceDeployment**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment
Note	Signal-based configuration settings for a ServiceInterface from which the content will be transmitted in the signal-based way over a communication medium. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInterfaceDeployments
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceDeployment*, *UploadablePackageElement*

▽

△

Class	SignalBasedServiceInterfaceDeployment			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 10.1: SignalBasedServiceInterfaceDeployment

10.3 Signal-To-Service Mapping

This chapter describes the mapping of `ServiceInterface` elements of a specific `AdaptivePlatformServiceInstance` to `ISignalTriggering`s.

Note that according to [TPS_MANI_03555] the same Ethernet socket (via `ISignalTriggering`, `PduTriggering`, and `SocketConnection`) may be used for signal-based and service-oriented communication at the same time.

Class	ServiceInstanceToSignalMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class represents a list of mappings of ServiceInstances to ISignalTriggerings. **Tags:** atp.Status=draft atp.recommendedPackage=ServiceInstanceToSignalMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
serviceInstanceToSignalMapping	ServiceInstanceToSignalMapping	*	aggr	This is one particular mapping accociation of a Service Instance to a number of ISignalTriggerings, **Tags:** atp.Status=draft

Table 10.2: ServiceInstanceToSignalMappingSet

Class	ServiceInstanceToSignalMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class is defined for a specific ServiceInstance and contains the mappings of elements of a ServiceInterface for which the ServiceInstance is defined to individual ISignalTriggerings. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
eventElementMapping	SignalBasedEventElementToISignalTriggeringMapping	*	aggr	Mapping of an event or an element inside of the event to an ISignalTriggering. **Tags:** atp.Status=draft
fieldMapping	SignalBasedFieldToISignalTriggeringMapping	*	aggr	Mapping of a field to ISignalTriggerings. **Tags:** atp.Status=draft
methodMapping	SignalBasedMethodToISignalTriggeringMapping	0..1	aggr	Mapping of a method to ISignalTriggerings. **Tags:** atp.Status=draft

▽

△

Class	ServiceInstanceToSignalMapping			
serviceInstance	AdaptivePlatformServiceInstance	0..1	ref	Reference to a ServiceInstance from which the corresponding ServiceInterface elements will be transported in the signal-based way over a communication medium. **Tags:** atp.Status=draft

Table 10.3: ServiceInstanceToSignalMapping

The ServiceInstanceToSignalMapping refers to an AdaptivePlatform-ServiceInstance and thereby defines which serviceInterface elements will be mapped by the aggregated eventElementMapping, methodMapping and/or fieldMapping to ISignalTriggerings. This is described in details in the following chapters.

10.3.1 SignalBasedEvent Mapping

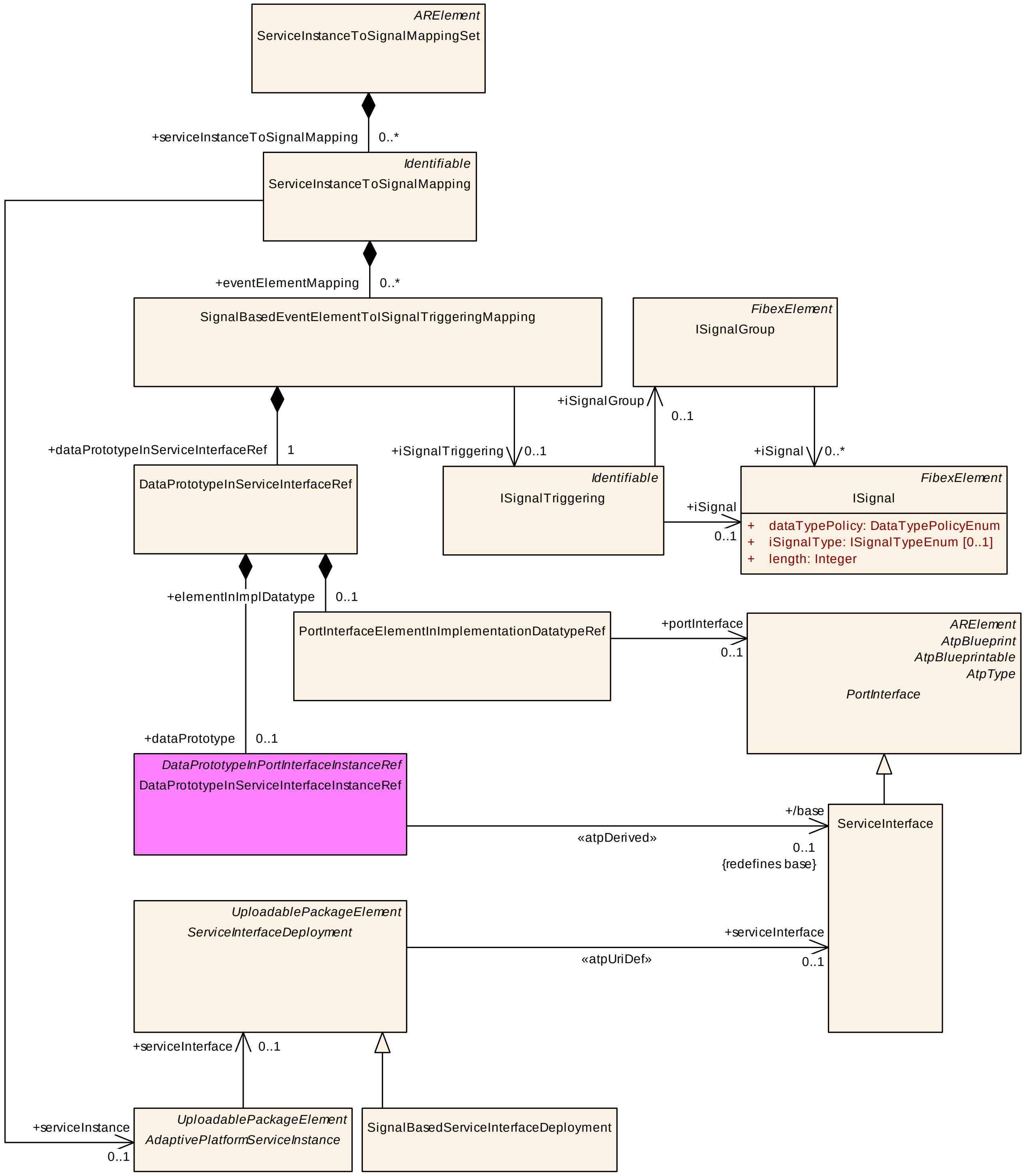


Figure 10.3: Mapping of Event elements to ISignals

[TPS_MANI_03124]{DRAFT} **`ServiceInterface.event` to `ISignalTriggering` mapping** ⌈ The `SignalBasedEventElementToISignalTriggeringMapping` meta-class provides the ability to map a `DataPrototype` defined in the context of a ServiceInterface to one ISignalTriggering of the ISignal or ISignalGroup. ⌋(RS_MANI_00029)

Class	SignalBasedEventElementToISignalTriggeringMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface event or an element that is defined inside of the event in case that the datatype is composite to an ISignalTriggering. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataPrototypeInServiceInterfaceRef	DataPrototypeInServiceInterfaceRef	1	aggr	Reference to a DataPrototype or to an internal structure of a DataPrototype in the context of a ServiceInterface. **Tags:** atp.Status=draft
iSignalTriggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport a piece of data of an event that is defined in a ServiceInterface in a signal-based way over a communication channel. **Tags:** atp.Status=draft

Table 10.4: SignalBasedEventElementToISignalTriggeringMapping

In the example sketched in Figure 10.4 the *TestEvent* in the *TestServiceInterface* is of type *struct1* that consists of a primitive element *a* and struct *b*. The struct *b* consists of the primitive elements *x*, *y* and *z*.

One ServiceInstanceToSignalMapping with several SignalBasedEventElementToISignalTriggeringMappings is necessary to map the *TestEvent* to the corresponding ISignalTriggerings.

One SignalBasedEventElementToISignalTriggeringMapping maps the DataPrototype that represents the *TestEvent* to an ISignalTriggering of an ISignalGroup. Here the dataPrototype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype according to the rules defined in [TPS_MANI_01136] and [TPS_MANI_01137].

Additional SignalBasedEventElementToISignalTriggeringMappings are necessary to map the primitive DataPrototypes *a*, *x*, *y* and *z* to ISignalTriggerings of ISignals located in the ISignalGroup.

The example shows the mapping of *z* to the ISignalTriggering. Here the elementInImplDatatype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype since it refers to the internal structure of an AutosarDataPrototype which is typed by a CppImplementationDataType. The context of this reference is defined by the *TestEvent* and struct *b*.

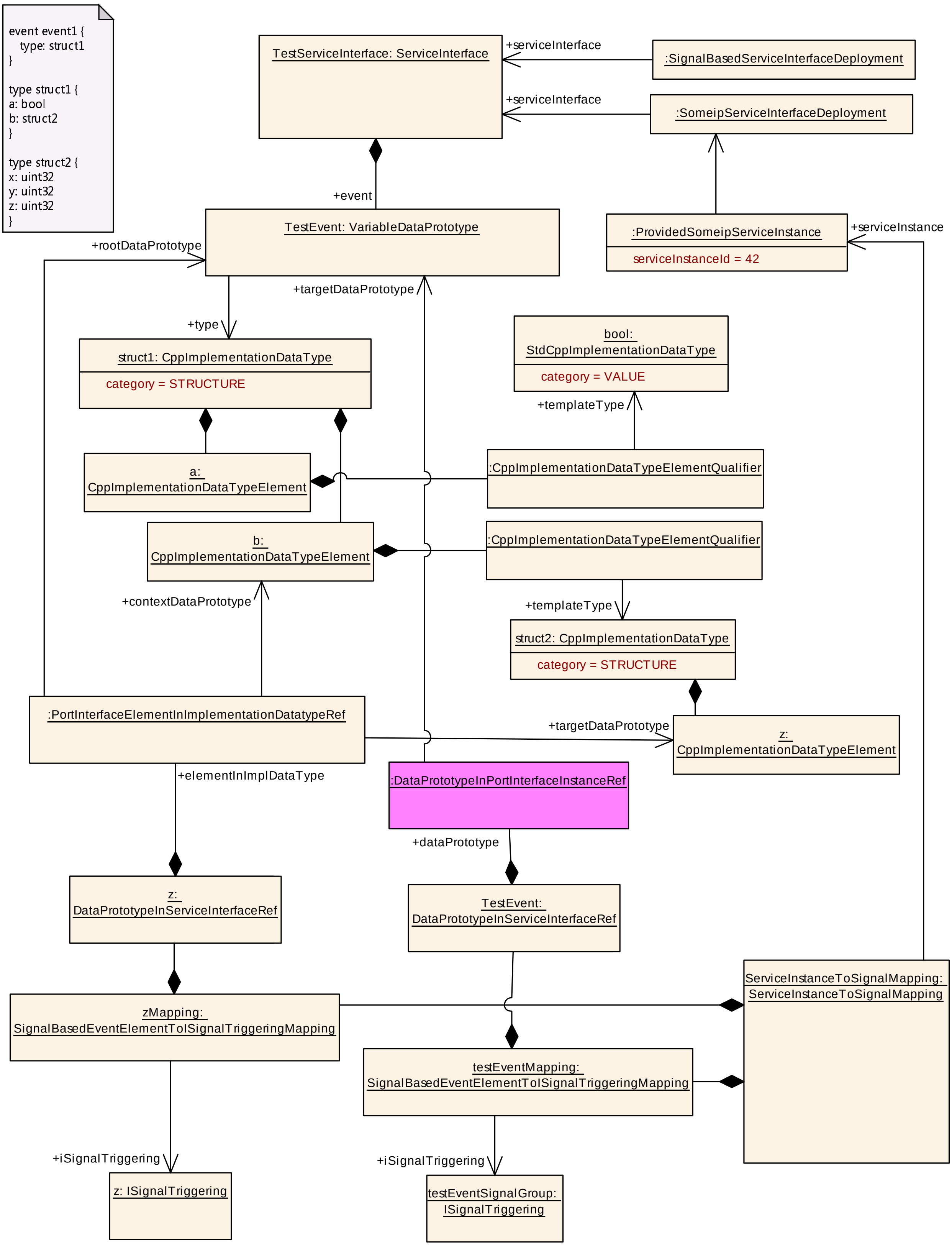


Figure 10.4: Example for a mapping of event content to a Signal

10.3.2 SignalBasedField Mapping

[TPS_MANI_03126]{DRAFT} **ServiceInterface.field mapping to ISignal-Triggerings** ⌈ The SignalBasedFieldToISignalTriggeringMapping meta-class provides the ability to map a field

- to one ISignalTriggering for the ISignalGroup representing the Notifier or
- to one ISignalTriggering for the ISignal representing the primitive Notifier element or
- to one ISignalTriggering for the ISignal representing the Getter-Call and one ISignalTriggering for the ISignal representing the Getter-Return or
- to one ISignalTriggering for the ISignal representing the Setter-Call and one ISignalTriggering for the ISignal representing the Setter-Return.

⌋*(RS_MANI_00029)*

It means that several SignalBasedFieldToISignalTriggeringMappings may be necessary to map a field to the corresponding ISignalTriggerings.

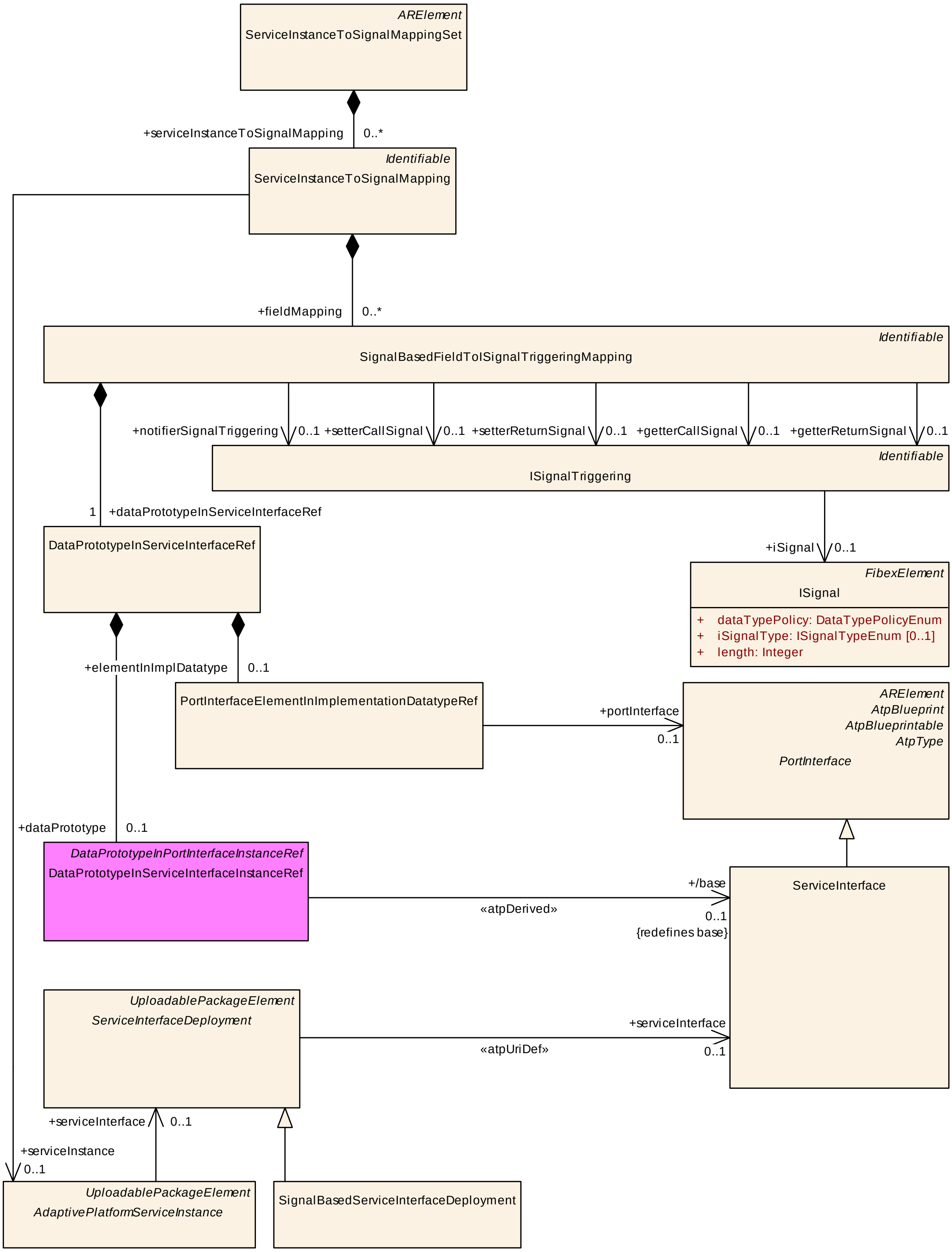


Figure 10.5: Mapping of Fields to ISignals

Class	**SignalBasedFieldToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface field to ISignalTriggerings that represent the notifier elements, the getter call and response, the setter call and response on a signal-based communication channel. . **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataPrototypeInServiceInterfaceRef	DataPrototypeInServiceInterfaceRef	1	aggr	Reference to a DataPrototype or to an internal structure of a DataPrototype in the context of a ServiceInterface. **Tags:** atp.Status=draft
getterCallSignal	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the getter method call in a signal-based way over a communication channel. **Tags:** atp.Status=draft
getterReturnSignal	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the getter method response in a signal-based way over a communication channel. **Tags:** atp.Status=draft
notifierSignalTriggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport a piece of data of a notifier in a signal-based way over a communication channel. **Tags:** atp.Status=draft
setterCallSignal	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the setter method call in a signal-based way over a communication channel. **Tags:** atp.Status=draft
setterReturnSignal	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the setter method response in a signal-based way over a communication channel. **Tags:** atp.Status=draft

Table 10.5: SignalBasedFieldToISignalTriggeringMapping

In the example sketched in Figure 10.6 the *testField* in the *testServiceInterface* is of type *struct1* that consists of the primitive elements *a* and *b*. The *testField* defines a notifier and a setter method.

One `SignalBasedFieldToISignalTriggeringMapping` maps the *TestField* to `ISignalTriggering`s for the Setter-Call and Setter-Return. Here the `dataPrototype` role of `DataPrototypeInServiceInterfaceRef` is used to refer to the `targetDataPrototype` (`field`) according to the rules defined in [TPS_MANI_01136] and [TPS_MANI_01137].

Additional `SignalBasedFieldToISignalTriggeringMapping`s are necessary to map the field notifier to the corresponding `ISignalTriggering`s. One `SignalBasedFieldToISignalTriggeringMapping` maps the *TestField* to the `ISignalTriggering` of the `ISignalGroup` that collects all `ISignal`s that transport the content of the notifier.

The primitive `DataPrototype`s *a* and *b* are mapped by additional `SignalBasedFieldToISignalTriggeringMapping`s to `ISignalTriggering`s of `ISignal`s located in the ISignalGroup. Here the elementInImplDatatype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype since it refers to the internal structure of an AutosarDataPrototype which is typed by a CppImplementationDataType. The context of this reference is defined by the *testField*.

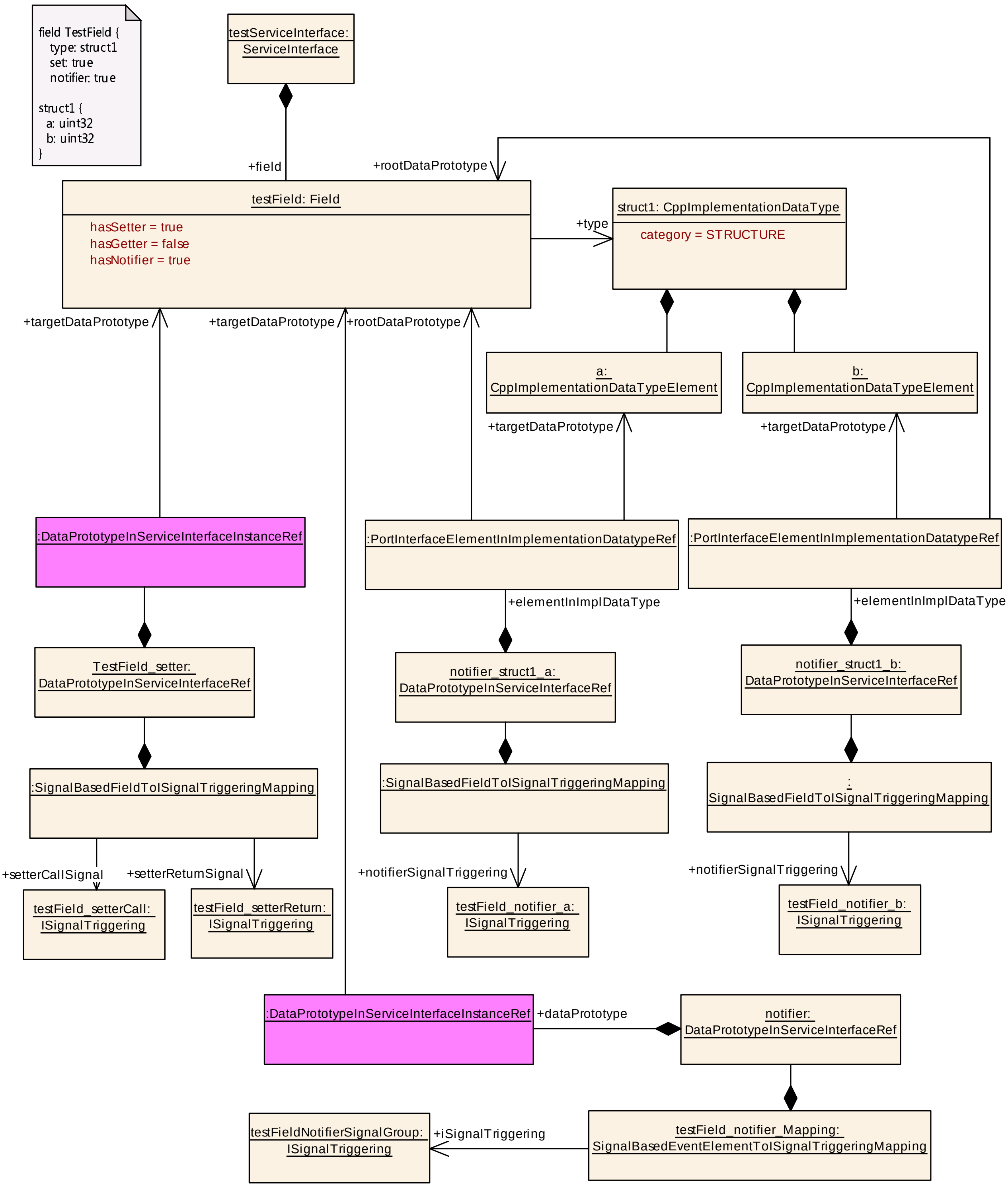


Figure 10.6: Example for a mapping of a field to Signals

10.3.3 SignalBasedMethod Mapping

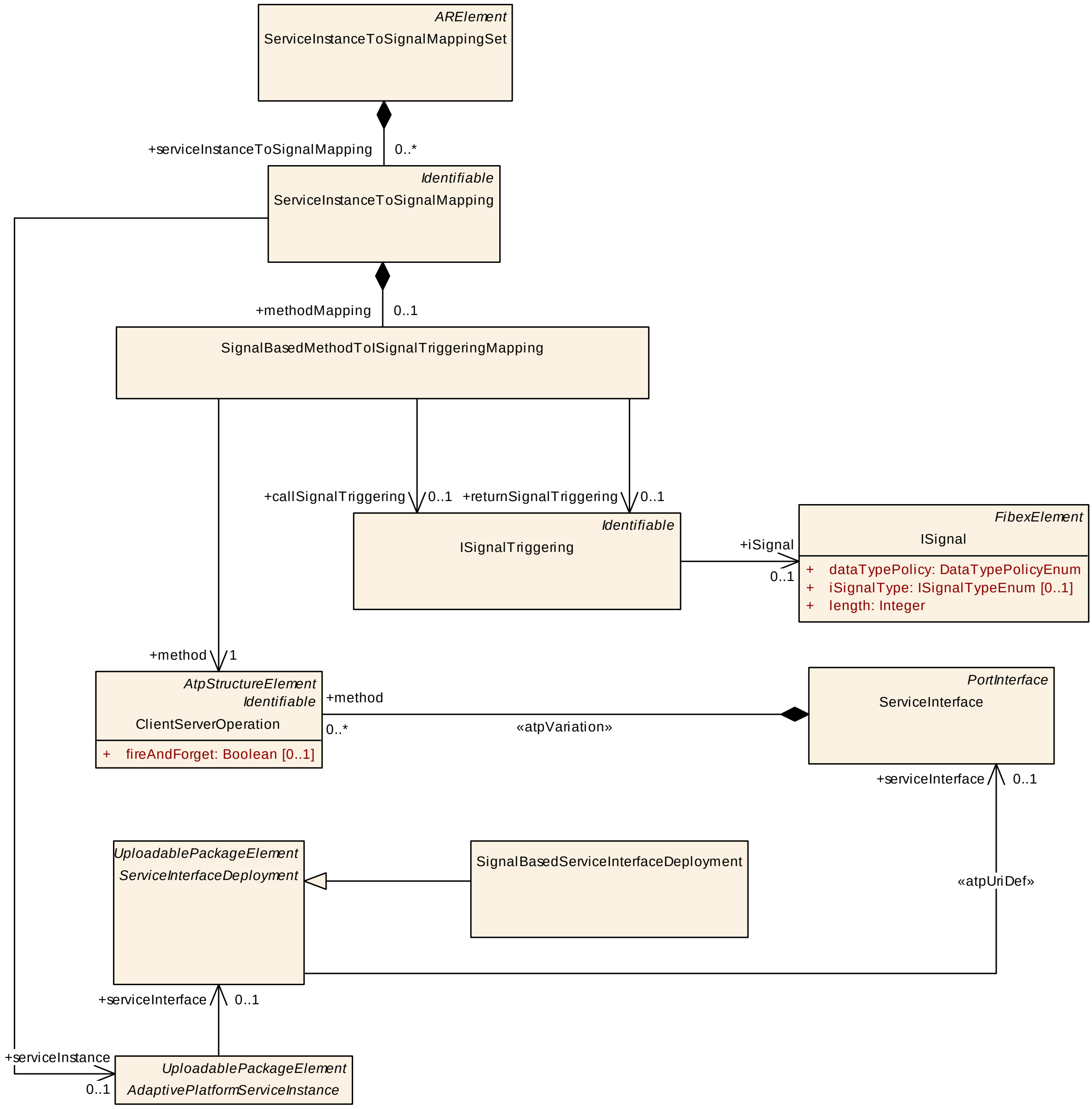


Figure 10.7: Mapping of Methods to ISignals

[TPS_MANI_03125]{DRAFT} **ServiceInterface.method to ISignalTriggerings mapping** ⌈ The SignalBasedMethodToISignalTriggeringMapping meta-class provides the ability to map a method to one ISignalTriggering for the ISignal representing the Method-Call and one ISignalTriggering for the ISignal representing the Method-Return. ⌋*(RS_MANI_00029)*

Class	SignalBasedMethodToISignalTriggeringMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface method to a ISignalTriggering. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
callSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the method call in a signal-based way over a communication channel. **Tags:** atp.Status=draft
method	ClientServerOperation	1	ref	Reference to a method defined in the context of a Service Interface. **Tags:** atp.Status=draft
returnSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the method response in a signal-based way over a communication channel. **Tags:** atp.Status=draft

Table 10.6: SignalBasedMethodToISignalTriggeringMapping

Please note that the `SignalBasedMethodToISignalTriggeringMapping` shall also be used for the mapping of `method`s where the value of attribute `method.fireAndForget` is set to `true`. In this case, only the `callSignalTriggering` shall be used since in the fire and forget Message Exchange Pattern only one message is sent from the service consumer to the service provider.

In the example sketched in Figure 10.8 the *Calibrate* method in the *TestServiceInterface2* is mapped with a single `SignalBasedMethodToISignalTriggeringMapping` to `ISignalTriggering`s for the Call and Return.

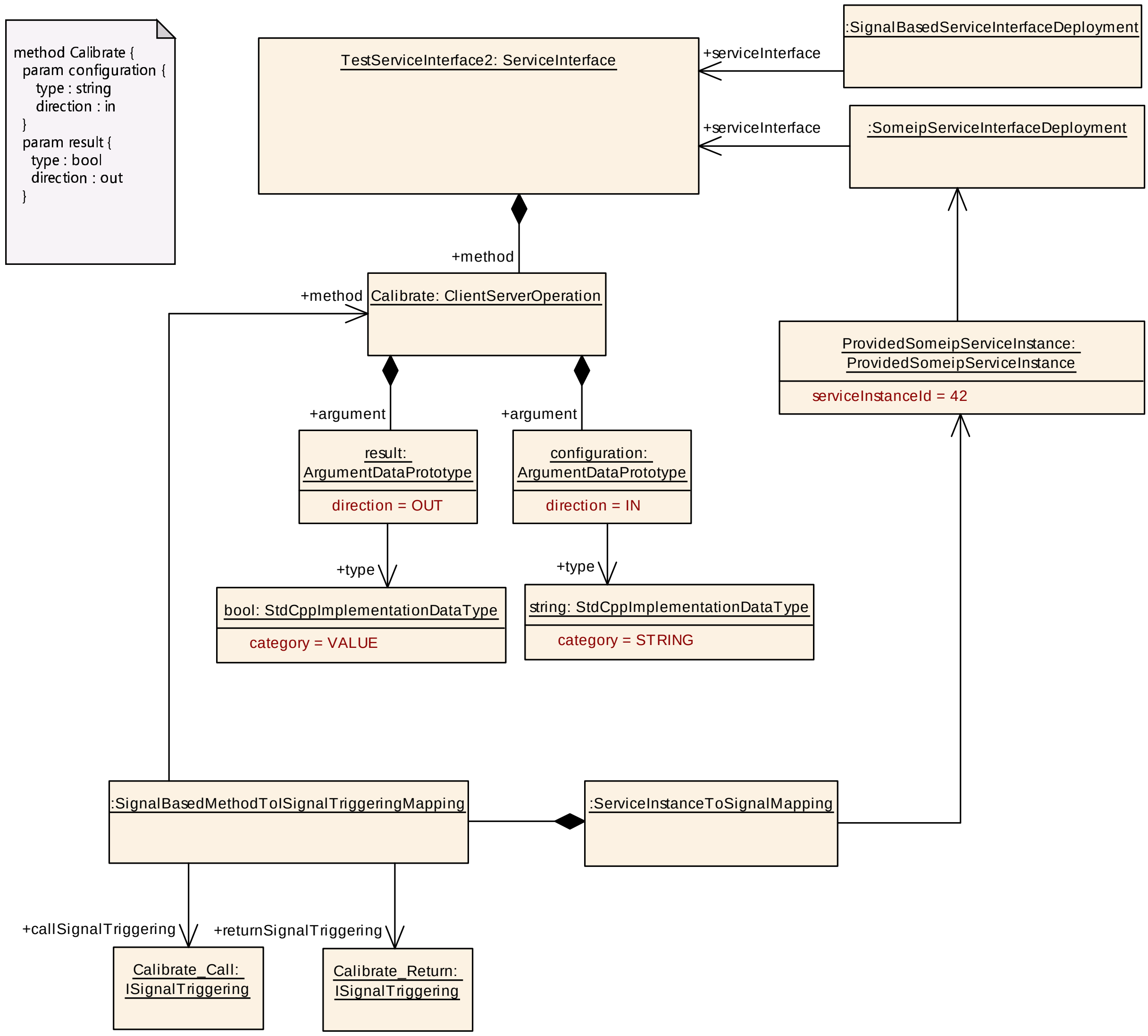


Figure 10.8: Example for a mapping of a method to Signals

11 REST

11.1 REST Design

11.1.1 Overview

Important note: the AUTOSAR SWS REST [5] defines a low-level API for REST-based communication. The content of this chapter, on the other hand, applies for the configuration of a not-yet standardized API on top of the ara::rest API.

In line with the target application domains of the *AUTOSAR adaptive platform* it can be expected that software will have use case to interact with generic web services inside and outside the vehicle.

Obviously, the communication partners need to agree on the applied communication approach to make this happen.

In other words, while it would be technically feasible to implement web services based on the existence of ServiceInterfaces it is still not very likely to happen for services that are completely outside the typical automotive domain and which have no incentive to embrace the communication approach of the *AUTOSAR adaptive platform*.

Therefore, the only viable option seems to extend the communications capabilities of the adaptive AUTOSAR stack to talk to web services in their "native language".

The conclusion to adopt web service communication approach does not only extend to the actual communication and transport conventions but also affects the way how information conveyed between a vehicle and a web service is described.

In order to fully implement a communication paradigm for information exchange with web services, the *AUTOSAR adaptive platform* needs to adopt conventions of data description that are typically supported by web services.

As a matter of fact, web services don't dive into data semantics nearly as deep as this is done in a typical automotive software and therefore seamlessly supported by the AUTOSAR meta-model. Consequently, AUTOSAR needs to define an alternative approach to data definition that matches with the conventions established for web services.

Consequently, the approach to define ApplicationDataTypes and their ImplementationDataType counterparts is not applicable for this case.

But still, the general AUTOSAR approach to structure application software into the definition of ApplicationSwComponentTypes that interact with the outside world via the existence of aggregated PortPrototypes applies also for software that interacts with web services.

In other words, interaction with web services need to be placed on the definition of a specific subclass of PortInterface in order to conform to the above mentioned statement.

The concrete definition of such a subclass of PortInterface requires a more specific understanding of the typical interaction patterns of web services.

While it is safe to conclude that the web breeds new technologies on nearly a weekly basis, there is still some stable core on which the modeling in AUTOSAR can rely on.

This stable core onto which the AUTOSAR modeling approach shall be based has been identified as the so-called "**Representational State Transfer**" [27] (a.k.a. REST) pattern.

Fundamentally, the REST approach requires a stateless communication among server and client, i.e. only data can be communicated.

The call of a method or operation (which is otherwise supported by means of the ServiceInterface or ClientServerInterface) is expressly out of scope.

[TPS_MANI_01103]{DRAFT} **Three-level approach to REST modeling** ⌈ The conversion of the REST pattern, as far as modeling is concerned, into AUTOSAR assumes a three-level structure:

Service This level represents the definition of an entire REST service.

In the AUTOSAR meta-model, this level is represented by meta-class RestServiceInterface.

Resource This level represents a resource in the context of the service. A resource can be used to structure the content of a service according to a given conceptual understanding of the semantics of the service.

For example, if a *sound mixer* were a service then it could make sense to define *audio source*, *output device*, etc as resources of the service. There can still be several sources and several output devices.

In the AUTOSAR meta-model, the resource level is represented by meta-class RestResourceDef.

Element The final level represents the definition of actual data with properties in the context of a resource. In the context of the above mentioned example of a *sound mixer* the element level of the *output device* resource could be populated by *volume*, *volume step-size*, *status*, etc.

In the AUTOSAR meta-model, the element level is represented by meta-class RestElementDef.

⌋*(RS_MANI_00033)*

The three-level approach described in [TPS_MANI_01103] is depicted in Figure 11.1.

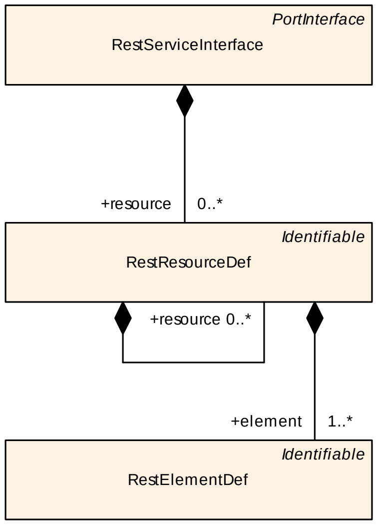


Figure 11.1: Big picture to REST modeling

Rest services are identified by means of a URI. The details of how the URI is created for a specific REST service can (because of the possibility of multiple instantiation of SwComponentTypes that aggregate PortPrototypes typed by a RestServiceInterface) only be resolved in the deployment phase where the specific instances are known.

The details of what makes a URI for a REST service as well as a description of how elements of the URI are sourced can be found in section 11.2.

Please note that in the domain of web services a service description is often provided in JSON format. The description of REST services in this chapter introduces the description of REST services to AUTOSAR and this has the consequence that ARXML has to be used for this purpose.

However, AUTOSAR does not oblige the usage of ARXML on the target platform, it only says that there shall be a point in time where the final model has to be available as ARXML and that exchange of AUTOSAR models shall only be done in ARXML format.

From the point of finalization going forward, proprietary conversions into whatever format for the sole purpose of uploading to a target platform is permitted.

Conversely, it is totally conceivable to create a conversion tool that takes an existing service description in JSON format and converts it into the ARXML representation described in this chapter.

Please note further that REST typically supports a filtering of information on the server, i.e. the client can apply a filter to only obtain the part of information on the server that passes the filter.

This filtering approach fully happens at run-time, there is no need to configure anything in the model in order to support the filtering of information on the server.

11.1.2 REST Service Interface

As depicted in Figure 11.2, RestServiceInterface is derived from PortInterface and can therefore be taken to type a PortPrototype.

In other words, the definition of a REST service creates a binding contract for the implementation of the ApplicationSwComponentType that aggregates a PortPrototype typed by a RestServiceInterface.

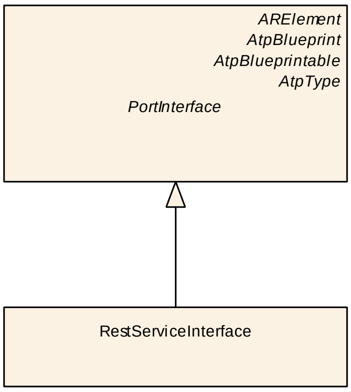


Figure 11.2: Modeling of the REST service

[TPS_MANI_01105]{DRAFT} **Semantics of RestServiceInterface** ⌈ A PortPrototype used to interact by means of the REST pattern with a web service shall be typed by RestServiceInterface. ⌋*(RS_MANI_00033)*

Class	**RestServiceInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents a REST service. **Tags:** atp.Status=draft atp.recommendedPackage=RestServiceInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
resource	RestResourceDef	*	aggr	This aggregation represents the collectioin of resources owned by the enclosing REST service. **Tags:** atp.Status=draft

Table 11.1: RestServiceInterface

11.1.3 REST Resource

[TPS_MANI_01120]{DRAFT} **Recursive definition of RestResourceDef** ⌈ The definition of RestResourceDef supports the aggregation of other RestResourceDef. In other words, it is possible to create a nested definition of RestResourceDefs. ⌋*(RS_MANI_00033)*

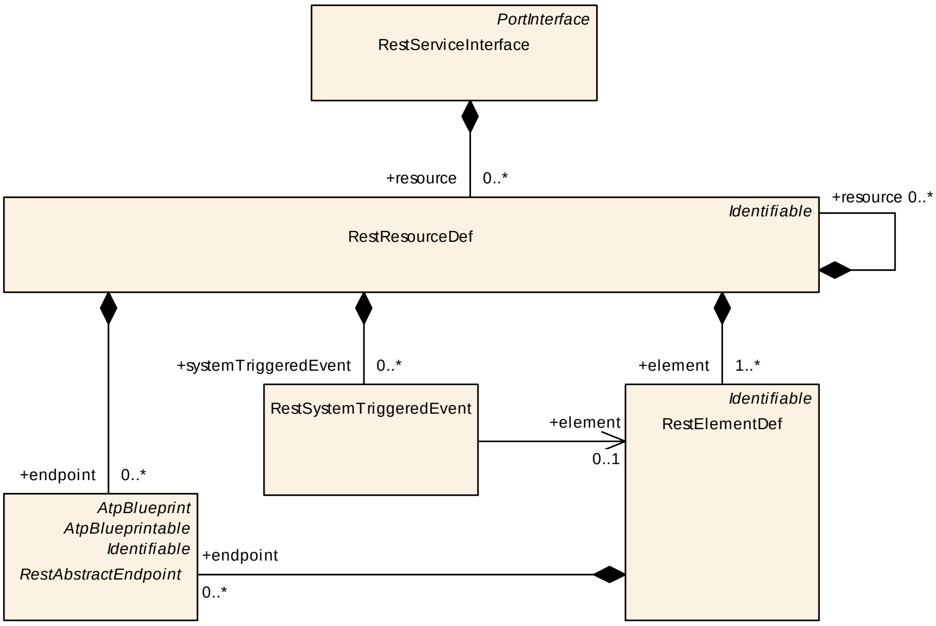


Figure 11.3: Modeling of the REST resource level

Class	**RestResourceDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents a resource inside a REST service. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
element	RestElementDef	1..*	aggr	This aggregation represents the elements of a resource. **Tags:** atp.Status=draft
endpoint	RestAbstractEndpoint	*	aggr	This aggregation represents the collection of endpoints on the resource level. **Tags:** atp.Status=draft
resource	RestResourceDef	*	aggr	This aggregation represenst the ability to create nested resource levels. **Tags:** atp.Status=draft
system TriggeredEvent	RestSystemTriggered Event	*	aggr	This represents the collection of system triggered events for the enclosing resource. **Tags:** atp.Status=draft

Table 11.2: RestResourceDef

[TPS_MANI_01121]{DRAFT} **Semantics of RestResourceDef.endpoint** ⌈ It is possible to define the API that shall be available for a specific RestResourceDef. For this purpose the aggregation of RestAbstractEndpoint in the role endpoint shall be used.

In particular the following concrete API elements (that directly correspond to the eponymous HTTP verbs) can be modeled:

GET For this purpose meta-class RestEndpointGet shall be used.

PUT For this purpose meta-class RestEndpointPut shall be used.

POST For this purpose meta-class RestEndpointPost shall be used.

DELETE For this purpose meta-class RestEndpointDelete shall be used.

⌋(*RS_MANI_00033*)

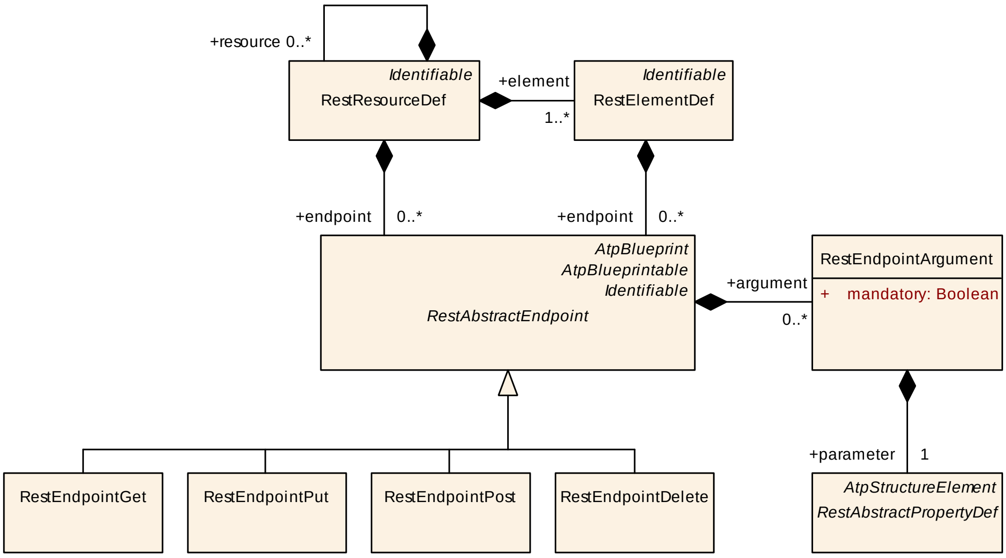


Figure 11.4: Modeling of the REST endpoints

[TPS_MANI_01122]{DRAFT} **Arguments to endpoints** ⌈ In many cases a concrete subclass of RestAbstractEndpoint needs arguments to fulfill its intended semantics. An argument to such an endpoint can be defined by means of the aggregation of RestEndpointArgument in the role RestAbstractEndpoint.argument . Arguments can be required to exist or may be optional. This question is clarified by means of attribute RestEndpointArgument.mandatory.

The actual "payload" of the argument is not defined by RestEndpointArgument itself, for this the aggregation RestEndpointArgument.parameter shall be used. ⌋ (*RS_MANI_00033*)

Class	***RestAbstractEndpoint*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class acts as a base class for the definition of endpoints within REST services. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	RestEndpointDelete, RestEndpointGet, RestEndpointPost, RestEndpointPut			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
argument	RestEndpointArgument	*	aggr	Some endpoints can require a list of arguments. **Tags:** atp.Status=draft

Table 11.3: RestAbstractEndpoint

Class	RestEndpointPut			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to model a REST endpoint with PUT semantics. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Rest AbstractEndpoint*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.4: RestEndpointPut

Class	RestEndpointGet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to model a REST endpoint with GET semantics. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Rest AbstractEndpoint*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.5: RestEndpointGet

Class	RestEndpointPost			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to model a REST endpoint with POST semantics. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Rest AbstractEndpoint*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.6: RestEndpointPost

Class	RestEndpointDelete			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to model a REST endpoint with DELETE semantics. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Rest AbstractEndpoint*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.7: RestEndpointDelete

Class	**RestEndpointArgument**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define an argument for a REST endpoint. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
mandatory	Boolean	1	attr	This attribute defines whether the argument is mandatory or whether it could be left out. **Tags:** atp.Status=draft
parameter	RestAbstractPropertyDef	1	aggr	This aggregation represents the concrete kind of argument to be used. **Tags:** atp.Status=draft

Table 11.8: RestEndpointArgument

[TPS_MANI_01123]{DRAFT} **System Triggered Event** ⌈ A `RestSystemTriggeredEvent` aggregated in the role `RestResourceDef.systemTriggeredEvent` can be modeled to indicate that a notifier for changes of the specific `RestElementDef` referenced in the role `RestSystemTriggeredEvent.element` shall be created.

By this means the server is able to inform any respectively configured client about changes of the referenced element. ⌋*(RS_MANI_00033)*

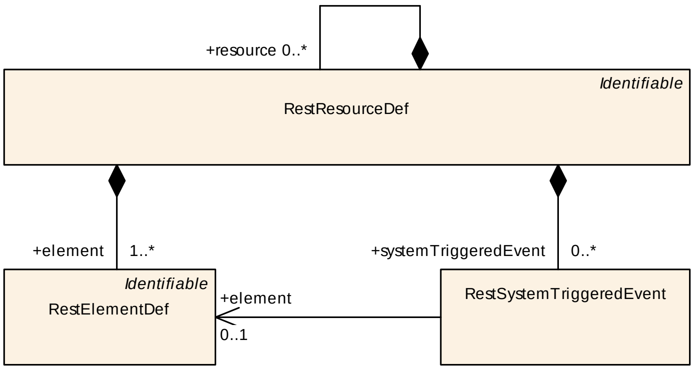


Figure 11.5: Modeling of the REST system triggered event

Class	**RestSystemTriggeredEvent**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to identify an element such that at runtime an event is generated when the value of the reference element changes. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	RestSystemTriggeredEvent			
element	RestElementDef	0..1	ref	This reference represent the element that is linked to the system triggered event. **Tags:** atp.Status=draft

Table 11.9: RestSystemTriggeredEvent

11.1.4 REST Element

[TPS_MANI_01124]{DRAFT} **Semantics of RestElementDef** ⌈ Meta-class RestElementDef represents the definition of data within a REST service. The specific definition of the data is done by way of aggregating so-called properties, i.e. RestElementDef aggregates RestAbstractPropertyDef in the role property. ⌋*(RS_MANI_00033)*

Class	**RestElementDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents an element of a resource that in turn is owned by a REST service. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
endpoint	RestAbstractEndpoint	*	aggr	This aggregation represents the definition of endpoints on the object level. **Tags:** atp.Status=draft
property	RestAbstractPropertyDef	1..*	aggr	This aggregation represents the collection of non-obligatory properties of the element level in a REST service. **Tags:** atp.Status=draft

Table 11.10: RestElementDef

Class	***RestAbstractPropertyDef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class acts as an abstract subclass for the definition of properties owned by the element level of a REST service definition. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	RestArrayPropertyDef, *RestPrimitivePropertyDef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.11: RestAbstractPropertyDef

As depicted by Figure 11.6, there is a certain variety of ways in which the properties of a REST element can be described.

However, the expressiveness of this description is in no way comparable to the richness of the semantics of an `ApplicationDataType` or a `CppImplementationDataType`.

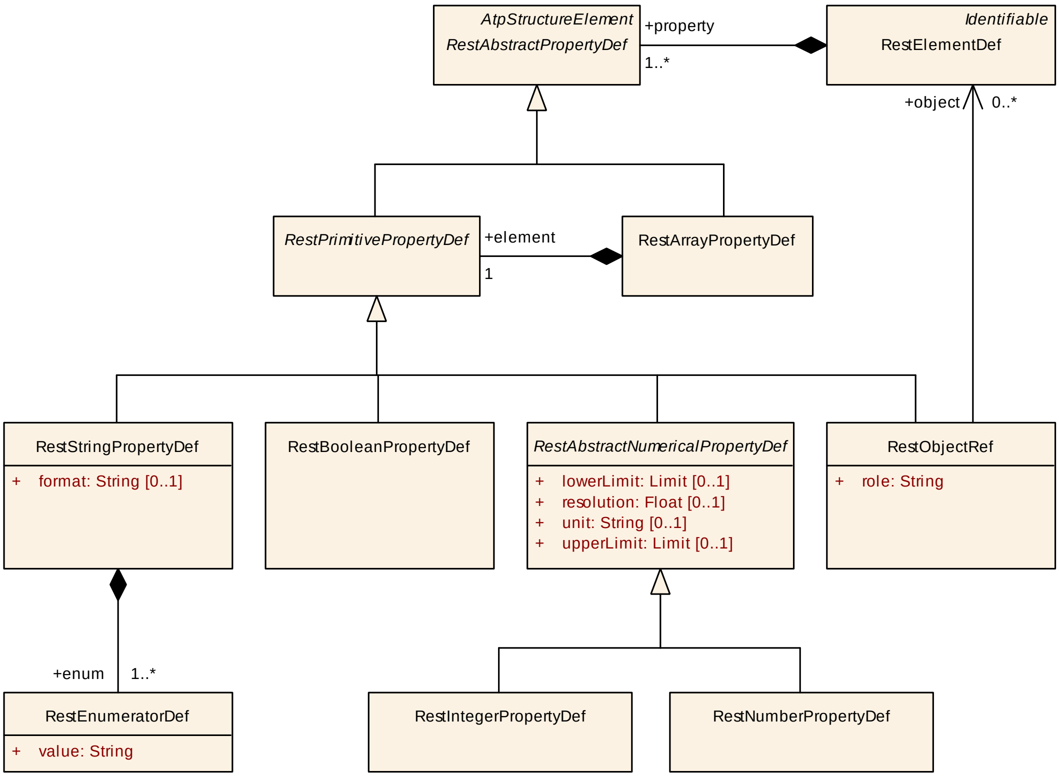


Figure 11.6: Modeling of the REST elements

[TPS_MANI_01125]{DRAFT} **Properties of REST elements can either be primitive or have array semantics** ⌈ The properties of REST elements can either be primitive or have array semantics.

There is no support for the creation of structures nor is the nesting of property definitions with array semantics supported.

This aspect is already clarified by the model (`RestArrayPropertyDef` directly aggregates `RestPrimitivePropertyDef`) and does not need to be expressed by a written constraint. ⌋*(RS_MANI_00033)*

Class	***RestPrimitivePropertyDef*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign
Note	This meta-class acts as an abstract base class for the definition of primitive properties of elements of a REST service. **Tags:** atp.Status=draft
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*

▽

△

Class	***RestPrimitivePropertyDef*** (abstract)			
Subclasses	*RestAbstractNumericalPropertyDef*, RestBooleanPropertyDef, RestObjectRef, RestStringPropertyDef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.12: RestPrimitivePropertyDef

Class	RestArrayPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a property of an element of a rest service where the property is supposed to represent an array of other primitive properties. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
element	RestPrimitiveProperty Def	1	aggr	This aggregation represents the definition of the base element type of the array property **Tags:** atp.Status=draft

Table 11.13: RestArrayPropertyDef

Class	RestBooleanPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property with boolean semantics. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*, *RestPrimitivePropertyDef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.14: RestBooleanPropertyDef

[TPS_MANI_01126]{DRAFT} **Definition of string properties** ⌈ Properties with string semantics can be defined by means of `RestStringPropertyDef`.

In many cases, the intention will be to only allow a certain number of values within the string property and define the potential values of the string property directly by the string property itself.

For this purpose, `RestStringPropertyDef` aggregates `RestEnumeratorDef` in the role `enum` that in turn allows for the definition of the predefined value by way of attribute `value`. ⌋*(RS_MANI_00033)*

Class	RestStringPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property with string semantics. **Tags:** atp.Status=draft			
Base	*ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, RestAbstractPropertyDef, RestPrimitivePropertyDef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
enum	RestEnumeratorDef	1..*	aggr	This aggregation represents the collection of enumerators for the enclosing string property. **Tags:** atp.Status=draft
format	String	0..1	attr	This attribute can be used to define a specific format that the value of the string property shall be conform with. **Tags:** atp.Status=draft

Table 11.15: RestStringPropertyDef

Class	RestEnumeratorDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define enumerator values that can be taken as a the value of the enclosing string property. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
value	String	1	attr	This attribute represents the ability to assign a value to an enumerator. **Tags:** atp.Status=draft

Table 11.16: RestEnumeratorDef

[TPS_MANI_01127]{DRAFT} **Limited support for data semantics in `RestAbstractNumericalPropertyDef`** ⌈ Meta-class `RestAbstractNumericalPropertyDef` allows for a limited support of data semantics by means of the following attributes:

`lowerLimit` This value represents a definition of the lower boundary of the allowed interval for this property. The value shall always be provided as a physical value.

`upperLimit` This value represents a definition of the upper boundary of the allowed interval for this property. The value shall always be provided as a physical value.

`unit` This value represents the unit of the property. It is only defied as a simple string without further formalization, i.e. it does not make use of `Unit` and/or `PhysicalDimension`.

`resolution` This attribute defines the resolution of the property. However, this definition should not be confused with a conversion into an internal value domain, comparable to the usage of `CompuMethod`. It just says that the value of the property shall have a certain resolution.

⌋*(RS_MANI_00033)*

For explanation, the values of a REST properties are typically conveyed from sender to receiver on top of a “JSON transport layer”. In other words, the serialization of the values ends up in a string-based format.

There is simply no need to define the conversion into a binary transport format that is used for typical automotive communication buses.

[TPS_MANI_01128]{DRAFT} **Difference between `RestIntegerPropertyDef` and `RestNumberPropertyDef`** ⌈ Both `RestIntegerPropertyDef` and `RestNumberPropertyDef` can benefit from the limited support for data semantics as described by [TPS_MANI_01127].

However, by design `RestIntegerPropertyDef` is foreseen to carry integer values while `RestNumberPropertyDef` is reserved for carrying non-integer[1] numbers. ⌋ *(RS_MANI_00033)*

Class	***RestAbstractNumericalPropertyDef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class acts as an abstract base class that contributes attributes for its subclasses that in turn represent a numerical property. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *RestAbstractPropertyDef*, *RestPrimitivePropertyDef*			
Subclasses	RestIntegerPropertyDef, RestNumberPropertyDef			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
lowerLimit	Limit	0..1	attr	This attribute specifies the lower limit of the property value. **Tags:** atp.Status=draft
resolution	Float	0..1	attr	This attribute specifies the resolution of a given value on a physical basis. **Tags:** atp.Status=draft
unit	String	0..1	attr	This attribute describes the lower limit of the property's value. **Tags:** atp.Status=draft
upperLimit	Limit	0..1	attr	This attribute describes the upper limit of the property's value. **Tags:** atp.Status=draft

Table 11.17: RestAbstractNumericalPropertyDef

Class	**RestIntegerPropertyDef**
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign
Note	This meta-class represents the ability to define a REST property with an integer semantics. **Tags:** atp.Status=draft

▽

[1] It would be inaccurate to describe these values as “float” because that would imply a certain representation in a binary layout in memory or on a bus. This binary format is not applicable in this case.

△

Class	RestIntegerPropertyDef			
Base	*ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, RestAbstractNumericalPropertyDef, RestAbstractPropertyDef, RestPrimitivePropertyDef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.18: RestIntegerPropertyDef

Class	RestNumberPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property with a numerical semantics. **Tags:** atp.Status=draft			
Base	*ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, RestAbstractNumericalPropertyDef, RestAbstractPropertyDef, RestPrimitivePropertyDef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 11.19: RestNumberPropertyDef

[TPS_MANI_01129]{DRAFT} **`RestObjectRef` is only needed for specific implementations of REST-based communication** ⌈ The existence of a `RestObjectRef` is only required for specific implementations of the REST-based communication approach.

The application of this reference has some pitfalls (it should only refer to elements in the same service, make sure to only reference the intended kind of element) and therefore needs to be applied carefully.

There is no formal support to make sure that only a certain kind of `RestElementDef` can be referenced. As a semi-formal support for the creation of references the attribute `RestObjectRef.role` has been introduced. It allows for the annotation of the kind of target `RestElementDef`. ⌋*(RS_MANI_00033)*

Class	RestObjectRef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDesign			
Note	This meta-class represents the ability to define a REST property that defines reference to another REST element. **Tags:** atp.Status=draft			
Base	*ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, RestAbstractPropertyDef, RestPrimitivePropertyDef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
object	RestElementDef	*	ref	This reference represents the ability to define constraints regarding the reference to another element, i.e. the reference identifies the element to which the reference is allowed to refer. **Tags:** atp.Status=draft

▽

△

Class	RestObjectRef			
role	String	1	attr	This attribute represents the ability to define a role for the reference to another element. **Tags:** atp.Status=draft

Table 11.20: RestObjectRef

The application of the attribute `RestObjectRef.role` is sketched in Figure 11.7. The example shows a REST service that makes heavy use of the referencing ability.

The roles (in *italics*) can be used for checking, i.e. the reference in the role *engine* should not point to e.g. a gastank object.

But again, this semantics - although the strongest that could be supported on M2 modeling level - is rather weak and may be subject to consistency problems.

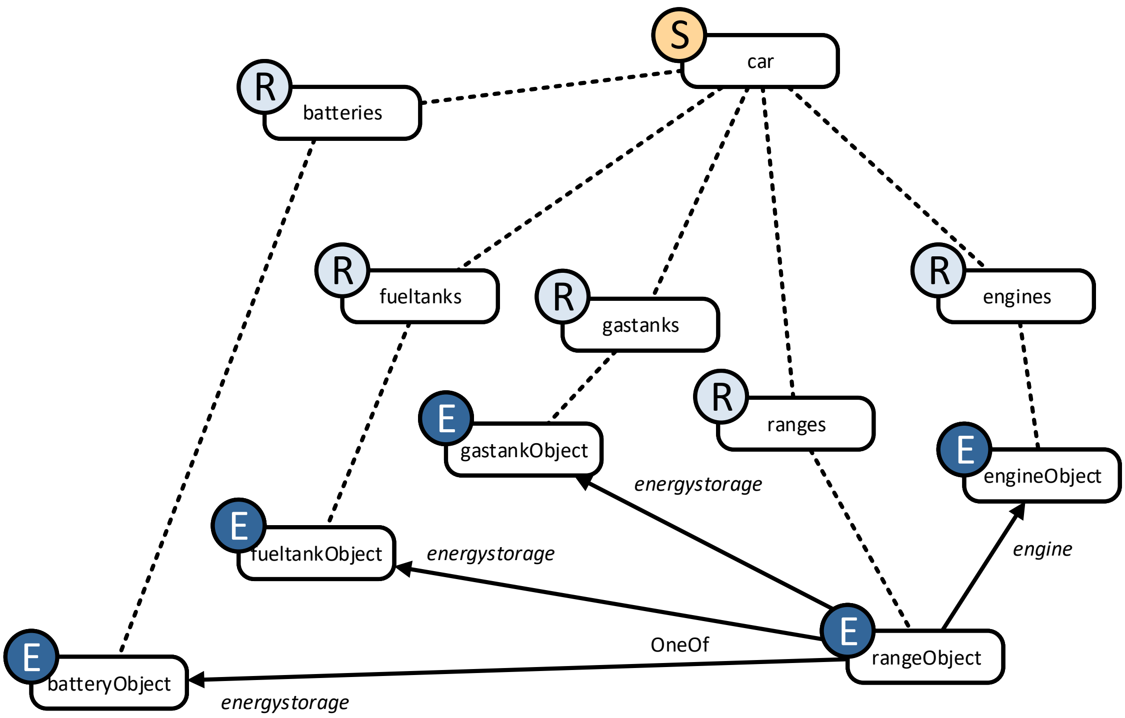


Figure 11.7: Example of the usage of the `role` attribute

11.2 REST Service Deployment

Important note: the AUTOSAR SWS REST [5] defines a low-level `API` for `REST`-based communication. The content of this chapter, on the other hand, applies for the configuration of a not-yet standardized `API` on top of the ara::rest `API`.

The ara::rest API requires fully-qualified URIs of the *remote communication end* to be passed to the various API elements. This is obviously a bad idea if application software should be kept independent of external resources.

Therefore, an API on top of ara::rest could focus on the path of the URI that is specific to the respective REST service formalized in a RestServiceInterface and inject the "non-portable" part of the URI of the *remote communication end* within an appropriately configured platform module.

Any approach for this purpose needs to take into account that software can be multiply instantiated (on different levels).

For example, the implementation of an Executable shall not make any assumptions about the number and/or behavior of the corresponding Processes launched.

This means that the URI may have elements used for the distinction of instances (created by launching the same Executable multiple times according to the definition of Processes in the execution manifest) of the same service.

To further drive this point home, Figure 11.8 has been created as a visualization of how a typical (i.e. it is assumed that RestResourceDef.resource does not exist to keep things simple) REST URI looks like.

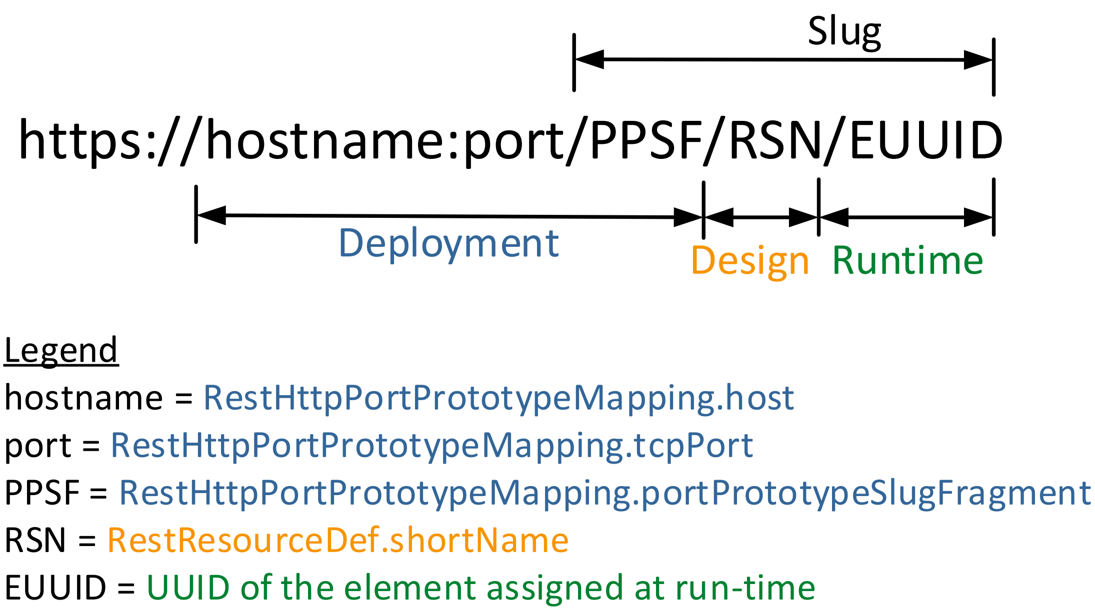


Figure 11.8: Structure of a typical URI for a REST service

As explained by Figure 11.8, the fully-qualified URI should be composed out of several ingredients contributed by different aspects of the configuration process.

The contribution from the design phase is described in section 11.1. The contribution from the deployment phase is depicted in Figure 11.9.

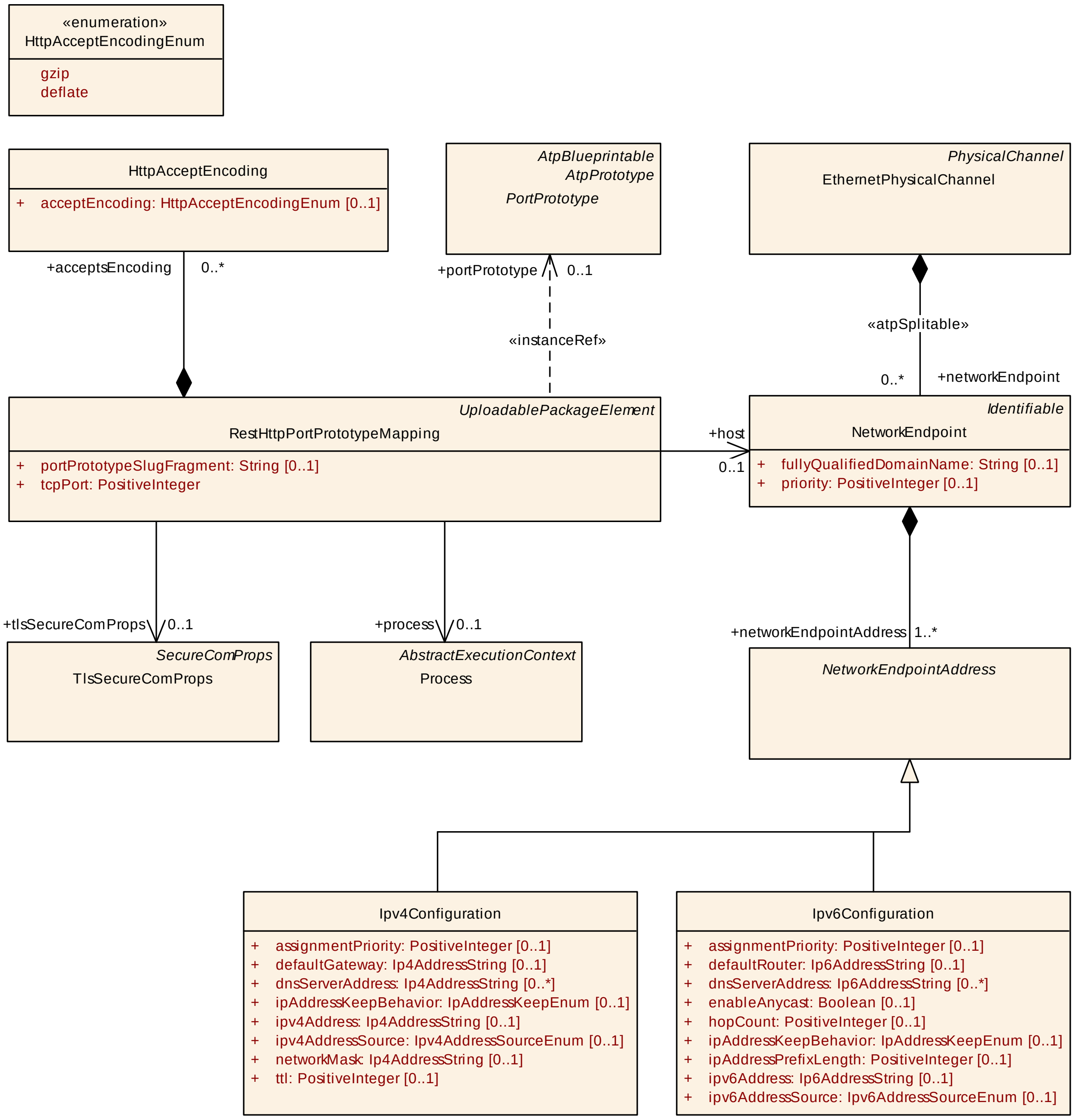


Figure 11.9: Modeling of the REST service deployment

In addition to the contributions from the design and deployment phase, some information that is only available at run-time when the objects that represents the data of a REST service are allocated in memory makes the list of ingredients for the creation of the URI of a REST service complete.

[TPS_MANI_01130]{DRAFT} **Structure of a typical URI for a REST service** ⌈ The part of the URI following the *hostname:port* tuple is usually called the slug.

In the case of a REST service the slug consists of three parts in the order listed below:

1. The representation of the **service instance** (that directly corresponds to the level of a PortPrototype) is contributed by the value of attribute RestHttpPortPrototypeMapping.portPrototypeSlugFragment. This part is defined on

deployment level in order to be sure that it is unique in the context to the *hostname:port* tuple.

2. The **resource** level within the slug is represented by the value of attribute RestResourceDef.shortName. This part is contributed on design level.
3. The identification of the **specific element** (on the level of RestElementDef) is represented by a UUID that is assigned at run-time.

⌋*(RS_MANI_00033)*

In other words, each URI represents a specific path within the tree structure rooted in the service level through levels of resources until finally the element level.

While [TPS_MANI_01130] defines the structure for the simplest and probably most like the most popular case (number of resource levels = 1) it is still necessary to understand the impact of more than one resource level on how the URI looks like.

This conclusion motivates the existence of [TPS_MANI_01131].

[TPS_MANI_01131]{DRAFT} **Impact of nested REST resources on the structure of REST URI** ⌈ The existence of RestResourceDef.resource results in the extension of the design contribution to the URI slug by additional levels consisting of the shortNames of the nested RestResourceDef aggregated in the role resource. ⌋ *(RS_MANI_00033)*

In other words, a specific path through the levels of aggregated RestResourceDefs represented by the respective shortNames, separated by '/' shall be inserted into the "RSN" slot depicted in Figure 11.8.

Please note that the rules for the creation of the slug of a REST URI are more or less arbitrary in terms of the usage of shortName from the model vs. a UUID assigned at run time.

It would technically be possible to use UUIDs instead of shortName on all levels, i.e. also for the "PPSF" and "RSN" slot.

However, this would dramatically decrease the readability of the URI and make it unnecessarily hard for human readers to understand the meaning of a given URI.

Class	**RestHttpPortPrototypeMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDeployment
Note	This meta-class represents the ability to define pieces of a URI for the REST service that cannot be contributed from the design point of view. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=RestHttpPortPrototypeMappings
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*

▽

△

Class	RestHttpPortPrototypeMapping			
Attribute	Type	Mul.	Kind	Note
acceptsEncoding	HttpAcceptEncoding	*	aggr	This aggrgation represents the collection of accepted encodings. **Tags:** atp.Status=draft
host	NetworkEndpoint	0..1	ref	This reference identifies the host configuration of the remote end. **Tags:** atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference identifies the instance of the PortPrototype to which the elements of the URI shall be defined. **Tags:** atp.Status=draft
portPrototypeSlugFragment	String	0..1	attr	This attribute contributes a string value to be taken as the slug reference that represents the PortPrototype level of a REST service. **Tags:** atp.Status=draft
process	Process	0..1	ref	This reference represents the process required for context of the mapping. **Tags:** atp.Status=draft
tcpPort	PositiveInteger	1	attr	This attribute represents the value of the TCP port applicable for this mapping. **Tags:** atp.Status=draft
tlsSecureComProps	TlsSecureComProps	0..1	ref	This represents the configuration of TLS applicable for the mapping. **Tags:** atp.Status=draft

Table 11.21: RestHttpPortPrototypeMapping

[TPS_MANI_01178]{DRAFT} **Semantics of RestHttpPortPrototypeMapping.acceptsEncoding** ⌈ The aggregation RestHttpPortPrototypeMapping.acceptsEncoding allows for a definition of the supported encodings from the client's perspective.

A client may support more than one encoding at the same time. Therefore the multiplicity of the aggregation has been set to 0..*. ⌋*(RS_MANI_00033)*

[constr_1569]{DRAFT} **Restriction for the scope of RestHttpPortPrototypeMapping.acceptsEncoding** ⌈ The attribute RestHttpPortPrototypeMapping.acceptsEncoding shall only be defined on the client side of a communication. ⌋*()*

[constr_1580]{DRAFT} **Restriction for the usage of RestHttpPortPrototypeMapping.acceptsEncoding** ⌈ Each member of HttpAcceptEncodingEnum shall only appear **at most** once in a particular RestHttpPortPrototypeMapping.acceptsEncoding. ⌋*()*

Please note that a preference rule for one encoding over others in the presence of more than one RestHttpPortPrototypeMapping.acceptsEncoding is subject to clarification in the respective SWS [5], see [SWS_REST_01834].

Class	**HttpAcceptEncoding**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDeployment			
Note	This meta-class represents the ability to specify the accept-encoding of an exchange using HTTP. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
acceptEncoding	HttpAcceptEncoding Enum	0..1	attr	This attribute is only used on the client side of the configuration for the purpose of stating the accepted compression algorithm.

Table 11.22: HttpAcceptEncoding

Enumeration	**HttpAcceptEncodingEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::REST::RESTDeployment
Note	This enumeration defines the value for the accept-encoding field of the HTTP header. **Tags:** atp.Status=draft
Literal	***Description***
deflate	Use deflate compression. **Tags:** atp.EnumerationValue=1
gzip	Use gzip pcompression. **Tags:** atp.EnumerationValue=0

Table 11.23: HttpAcceptEncodingEnum

12 Uploadable Software Package

12.1 Overview

One of the key features of the *AUTOSAR adaptive platform* is the ability to extend the software on a given ECU without having to reflash the entire ECU. Instead, software packages are uploaded to the ECU where the content is taken care of by responsible platform modules.

The reason why this topic is relevant for the modeling is the fact that an uploadable software package consists not only of software itself but also of manifest content required to support the integration of the uploaded software with the existing platform instance.

As far as the meta-model is concerned, the discussion about manifests and which manifest content needs to go with which other model elements doesn't care about the file granularity. In other words, it would not make sense to formalize the uploadable software package on the basis of references to files that carry model elements.

Instead, the view on the manifest topic from the modeling point of view focuses on model elements that make up manifest content.

Therefore, the modeling of an uploadable software package allows for putting references to all the required model elements that, in their entirety, make up the manifest of the corresponding application software that is also going to end up in the uploadable software package.

From the formal point of view, such an uploadable software package is modeled as a so-called SoftwareCluster. This meta-class is the root element that in turn describes all the necessary content of an uploadable software package.

However, the software package obviously isn't created out of thin air. It is the result of a workflow that starts from the formulation of requirements on the content of a SoftwareCluster.

These requirements are formalized by means of meta-class SoftwareClusterDesigns.

The relation between SoftwareClusterDesign and SoftwareCluster is depicted in Figure 12.1.

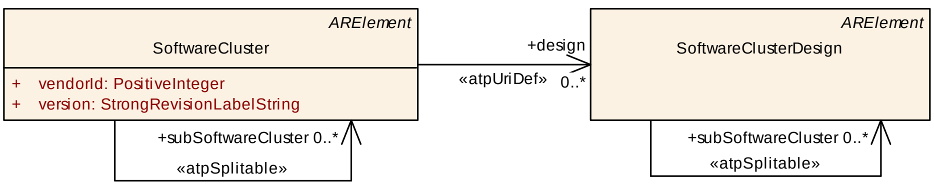


Figure 12.1: Relation of SoftwareClusterDesign to SoftwareCluster

[TPS_MANI_01109]{DRAFT} **Semantics of UploadablePackageElement** ⌈ In order to keep the complexity of the modeling of SoftwareCluster as low as possible abstract meta-class UploadablePackageElement has been created.

This allows for the referencing of model elements derived from UploadablePackageElement that need to be considered in an uploadable software package from within a SoftwareCluster with just the reference containedPackageElement.

The same applies for SoftwareClusterDesign and the respective reference requiredPackageElement. ⌋*(RS_MANI_00035)*

Class	***UploadablePackageElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class acts as an abstract base class for all meta-classes that need to be added to an uploadable software package in order to complete the manifest content. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractExecutionContext*, *AdaptivePlatformServiceInstance*, ApApplicationError, ApApplicationErrorDomain, ApApplicationErrorSet, DeterministicClient, E2EProfileConfigurationSet, *PersistencyDeployment*, PersistencyFile, PersistencyPortPrototypeToFileArrayMapping, PersistencyPortPrototypeToKeyValueDatabaseMapping, PlatformHealthManagementContribution, PortInterfaceToDataTypeMapping, ProcessToMachineMappingSet, RestHttpPortPrototypeMapping, *ServiceInstanceToMachineMapping*, ServiceInstanceToPortPrototypeMapping, *ServiceInterfaceDeployment*, StartupConfigSet, TimeSyncPortPrototypeToTimeBaseMapping			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 12.1: UploadablePackageElement

Please note that this approach to collecting elements is very similar in semantics to System.fibexElement or DiagnosticContributionSet.element.

12.2 Software Cluster Design

[TPS_MANI_01112]{DRAFT} **Semantics of SoftwareClusterDesign** ⌈ The existence of a SoftwareClusterDesign represents formalized requirements that have initially been formulated by an OEM and that may be enriched as the development of the software progresses.

Finally, the SoftwareClusterDesign shall be taken by the integration as a further input to the definition of the result of the integration step: the definition of the SoftwareCluster. ⌋*(RS_MANI_00035)*

Just to be sure, the SoftwareClusterDesign is not intended to be uploaded to the target platform. It is just an early form of the final SoftwareCluster that indeed gets uploaded. The existence of the SoftwareClusterDesign is motivated from the methodological point of view.

[constr_1557]{DRAFT} **Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category** ⌈ The AUTOSAR standard reserves the following values of attribute SoftwareClusterDesign.category and SoftwareCluster.category:

- ROOT_SOFTWARE_CLUSTER
- SUB_SOFTWARE_CLUSTER

⌋()

[TPS_MANI_01161]{DRAFT} **Impact of values of category on the semantics of SoftwareClusterDesign** ⌈ A SoftwareClusterDesign of category ROOT_SOFTWARE_CLUSTER may refer to other SoftwareClusterDesigns of category SUB_SOFTWARE_CLUSTER in the role subSoftwareCluster and thereby offer a way to further break down the creation of a SoftwareClusterDesign. ⌋ *(RS_MANI_00035)*

[constr_1558]{DRAFT} **Existence of SoftwareClusterDesign.diagnosticAddress** ⌈ The aggregation of SoftwareClusterDiagnosticAddress at SoftwareClusterDesign in the role diagnosticAddress shall only exist if the value of SoftwareClusterDesign.category is set to ROOT_SOFTWARE_CLUSTER. ⌋()

[constr_1559]{DRAFT} **Existence of SoftwareClusterDesign.subSoftwareCluster** ⌈ The Reference from SoftwareClusterDesign to itself in the role subSoftwareCluster shall only exist if the value of SoftwareClusterDesign.category is set to ROOT_SOFTWARE_CLUSTER. ⌋()

[constr_1560]{DRAFT} **Usage of SoftwareClusterDesign.requiredARElement** ⌈ The reference SoftwareClusterDesign.requiredARElement shall not be used to refer to another SoftwareClusterDesign or even SoftwareCluster. ⌋()

Rationale for the existence of [constr_1560]: dedicated references are defined for the purpose of referring to SoftwareClusterDesigns.

[TPS_MANI_01162]{DRAFT} **Semantics of SoftwareClusterDesign.dependsOn** ⌈ A SoftwareClusterDesign has the ability to refer to other SoftwareClusterDesigns in the role dependsOn.dependentSoftwareClusterDesign to express that the functionality of the referenced SoftwareClusterDesign is required to enable the full functionality of the referencing SoftwareClusterDesign.

Attribute SoftwareClusterDesignDependency.dependency allows for the definition of a **non-formal condition**. In other words, the dependency shall be applicable only if the condition is fulfilled. ⌋*(RS_MANI_00035)*

Both the reference in the role SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign as well as the reference in the role SoftwareClusterDesign.subSoftwareCluster factually define a dependency between SoftwareClusterDesigns.

However, the difference is that SoftwareClusterDesigns referenced in the role subSoftwareCluster may not have subSoftwareClusters themselves (as explained by [constr_1559]).

Class	SoftwareClusterDesignDependency			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class has the ability to support the expression of a dependency from one SoftwareCluster to another. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dependency	String	1	attr	This attribute allows for the definition of a non-formal dependency to the target SoftwareClusterDesign.
dependentSoftwareClusterDesign	SoftwareClusterDesign	0..1	ref	This reference identifies the dependent SoftwareClusterDesign. **Tags:** atp.Status=draft

Table 12.2: SoftwareClusterDesignDependency

[constr_1561]{DRAFT} **Existence of SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign** ⌈ Within the context of a specific SoftwareClusterDesign, the references SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign shall not refer to the same target SoftwareClusterDesign. ⌋*()*

Rationale for the existence of [constr_1561]: the existence of a reference to another SoftwareClusterDesign in two different roles creates ambiguity and also there is no definition for the semantics of such a scenario.

[TPS_MANI_01211]{DRAFT} **Specification of executable software within SoftwareClusterDesign** ⌈ One of the most prominent contents of an uploadable software package is the reference to the executable software.

Within the definition of a SoftwareClusterDesign, this reference is implicitly given by means of the reference SoftwareCluster.containedProcess.

The target of SoftwareClusterDesign.containedProcess is a ProcessDesign that represents the design-level representation of an instance (formalized as Process) of the corresponding executable program (the software image), formalized as Executable ⌋*(RS_MANI_00035)*

[TPS_MANI_01113]{DRAFT} **Semantics of SoftwareClusterDesign.diagnosticAddress** ⌈ The existence of the attribute SoftwareClusterDesign.diagnosticAddress can be used to express information about the distribution of diagnostic addresses even in a very early stage of development, i.e. this is typically done by an OEM.

This includes the ability to specify multiple (i.e. several functional plus one physical) diagnostic addresses, thus the multiplicity of diagnosticAddress is set to 0..*. ⌋ *(RS_MANI_00035)*

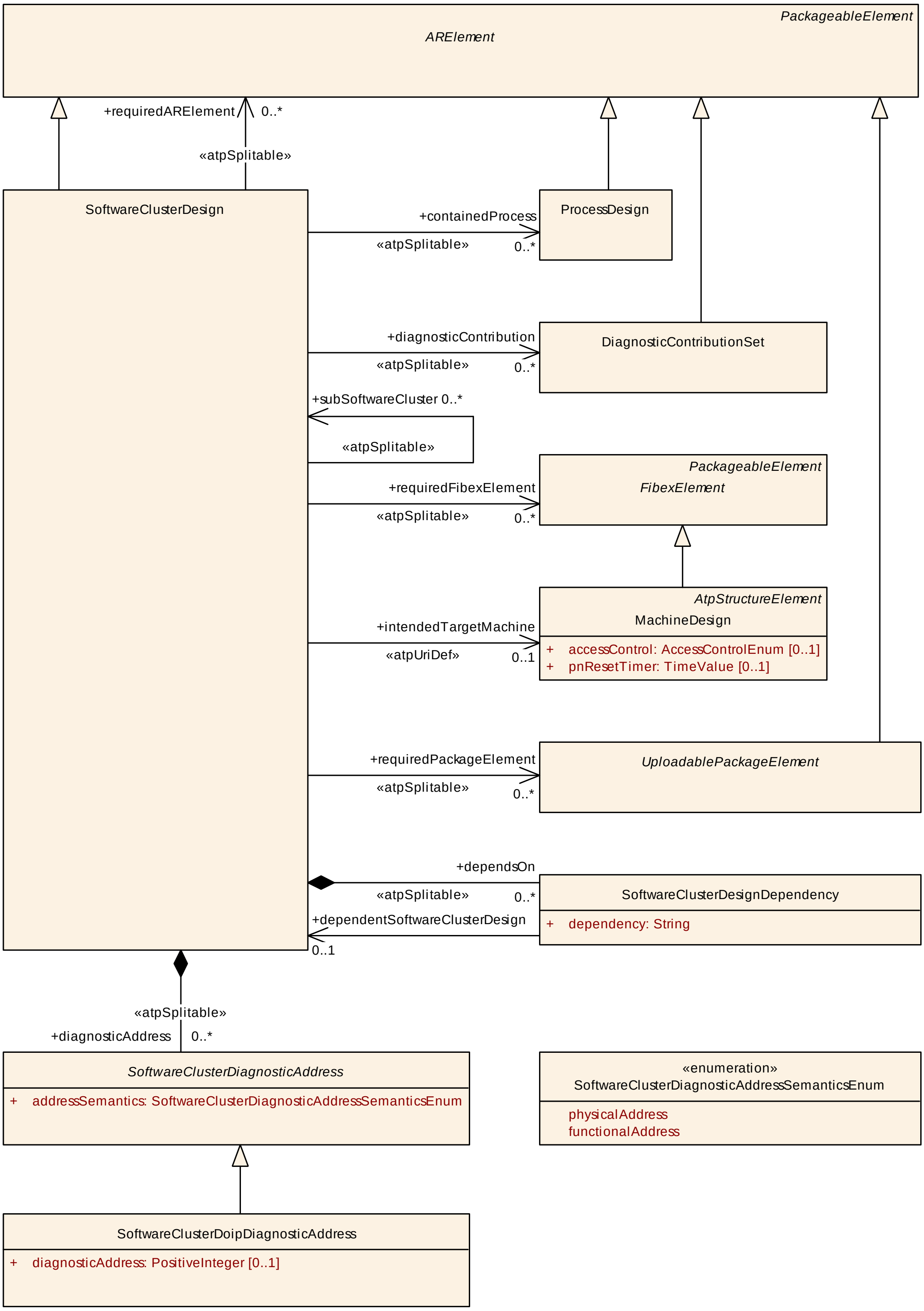


Figure 12.2: Modeling of SoftwareClusterDesign

Class	SoftwareClusterDesign			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability for the OEM to design the grouping of software uploadable to a specific target Machine. **Tags:** atp.Status=draft atp.recommendedPackage=SoftwareClusterDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
contained Process	ProcessDesign	*	ref	This reference represent the ProcessDesigns contained in the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedProcess atp.Status=draft
dependsOn	SoftwareClusterDesign Dependency	*	aggr	This aggregation allows for the specification of a dependency. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dependsOn atp.Status=draft
diagnostic Address	SoftwareCluster DiagnosticAddress	*	aggr	This aggregaton is used to specify the diagnsotic address. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticAddress atp.Status=draft
diagnostic Contribution	DiagnosticContribution Set	*	ref	This reference identifes the corresponding collection of DiagnosticContributionSet. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticContribution atp.Status=draft
intendedTarget Machine	MachineDesign	0..1	ref	This reference can be taken to identify the Machine Design for which the final SoftwareCluster shall be developed. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
required ARElement	ARElement	*	ref	This reference represents the collection of ARElements that are required for the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredARElement atp.Status=draft
requiredFibex Element	FibexElement	*	ref	This reference represents the collection of fibexElements that are required for the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredFibexElement atp.Status=draft
required Package Element	UploadablePackage Element	*	ref	This reference points to uploadable elements that have been identified as relevant in the context of the enclosing SoftwareClusterDesign. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredPackageElement atp.Status=draft

▽

△

Class	**SoftwareClusterDesign**			
subSoftware Cluster	SoftwareClusterDesign	*	ref	This reference is used to identify the sub-SoftwareCluster Designs of an "umbrella" SoftwareClusterDesign. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=subSoftwareCluster atp.Status=draft

Table 12.3: SoftwareClusterDesign

[TPS_MANI_01117]{DRAFT} **Semantics of SoftwareClusterDesign.intendedTargetMachine** ⌈ The specification of SoftwareClusterDesign.intendedTargetMachine allows for focusing the specification of an uploadable software package to a specific MachineDesign from early phases of a development project. ⌋ *(RS_MANI_00035)*

Please note that SoftwareCluster doesn't have a dedicated reference to the target Machine.

This relation is expressed by means of a reference to Process that in turn can be mapped to a dedicated Machine by means of a ProcessToMachineMapping. In this context, [constr_1536] applies.

[TPS_MANI_01118]{DRAFT} **Relation between SoftwareClusterDesign and DiagnosticContributionSet** ⌈ An important aspect of the definition of a SoftwareClusterDesign is the question what diagnostic extract shall be associated with the SoftwareClusterDesign.

For this purpose, a reference from SoftwareClusterDesign to DiagnosticContributionSet in the role diagnosticContribution is provided.

In an early stage of the development process, it is intentionally made possible to reference multiple DiagnosticContributionSets in order to support the decentralized (e.g. partly done by OEM and partly done by supplier) configuration of the diagnostics stack. ⌋ *(RS_MANI_00035)*

[TPS_MANI_01189]{DRAFT} **Software Cluster and DiagnosticContributionSet.category** ⌈ A DiagnosticContributionSet used in the context of a SoftwareCluster shall set the value of attribute category to DIAGNOSTICS_SWCL_EXTRACT. ⌋ *(RS_MANI_00035)*

[constr_1562]{DRAFT} **Existence of SoftwareClusterDesign.diagnosticContribution** ⌈ The existence of the reference SoftwareClusterDesign.diagnosticContribution is limited to SoftwareClusterDesigns where attribute category is set to the value ROOT_SOFTWARE_CLUSTER. ⌋ *()*

Rationale for the existence of [constr_1562]: the definition of the diagnostic behavior is limited to the root level of a structure of SoftwareClusterDesigns in the same spirit that caused the existence of [constr_1558].

Please mind the intentionally introduced difference between SoftwareCluster and SoftwareClusterDesign in terms of the relation to DiagnosticContributionSet.

In other words, the multiplicity of the references to DiagnosticContributionSet intentionally differ.

As already explained, the SoftwareClusterDesign shall support the decentralized configuration of the DiagnosticContributionSet while the SoftwareCluster requires the existence of a final (merged) DiagnosticContributionSet.

[TPS_MANI_01119]{DRAFT} **Reference to model elements from SoftwareClusterDesign** ⌈ SoftwareClusterDesign has the ability to define the following references to model elements relevant for the definition of an uploadable software package:

- references to meta-classes derived from UploadablePackageElement are formalized by way of SoftwareClusterDesign.requiredPackageElement.
- references to meta-classes derived from ARElement are formalized by way of SoftwareClusterDesign.requiredARElement.
- references to meta-classes derived from FibexElement are formalized by way of SoftwareClusterDesign.requiredFibexElement.

⌋*(RS_MANI_00035)*

Please note that the conversion of a SoftwareClusterDesign to a SoftwareCluster is not formalized by AUTOSAR. This step can be done by a tool at the discretion of the integrator.

In other words, in some cases it may be applicable to do this conversion relatively early in the development project while other projects may require to keep the SoftwareClusterDesign around for a longer period in time.

12.3 Software Cluster

12.3.1 Software Cluster General Modeling

[TPS_MANI_01110]{DRAFT} **Semantics of SoftwareCluster** ⌈ The existence of a SoftwareCluster represents an uploadable software package. ⌋*(RS_MANI_00035)*

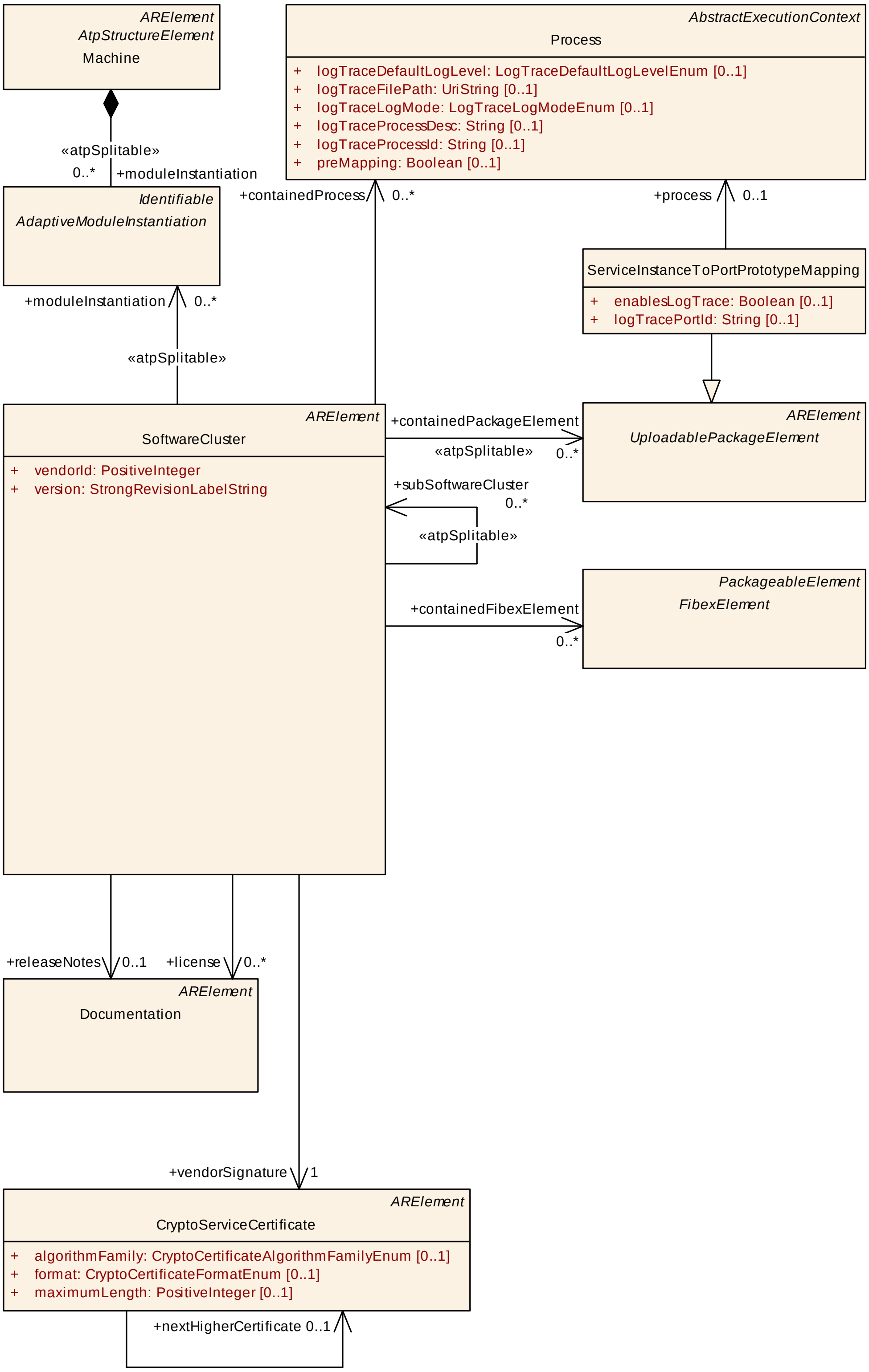


Figure 12.3: Modeling of SoftwareCluster

A SoftwareCluster comes with a semantic version expressed by means of the following **mandatory** components:

- Major version

- Minor version
- Patch version
- Build number

For the purpose of specifying version numbers, AUTOSAR defines the primitive class RevisionLabelString. The latter is in principle able to carry the required content. However, the build number is **optional** in the definition of the RevisionLabelString.

In response to this shortcoming, a further meta-class named StrongRevisionLabelString is defined that fulfills the requirement of carrying the list of version components as mandatory elements.

[TPS_MANI_01213]{DRAFT} **Semantics of meta-class StrongRevisionLabelString** ⌈ Meta-class StrongRevisionLabelString supports the specification of a version number that consists of four mandatory components:

- Major version
- Minor version
- Patch version
- Build number

⌋*(RS_MANI_00035)*

Primitive	**StrongRevisionLabelString**
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage
Note	This primitive represents a revision label which identifies an engineering object. It represents a pattern which requires four integer numbers separated by a dot, representing from left to right MajorVersion, MinorVersion, PatchVersion, and BuildVersion. Legal patterns are for example: 4.0.0.3456 4.0.0.1234565 **Tags:** atp.Status=draft xml.xsd.customType=STRONG-REVISION-LABEL-STRING xml.xsd.pattern=[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+ xml.xsd.type=string

Table 12.4: StrongRevisionLabelString

Please note that the build number does not necessarily have to be consecutively incremented between two builds. In some cases the build number is created by creating a hash over the build.

In such a case it would not make sense to include the build number in a greater/less comparison while a comparison for equality/inequality may positively make sense. This aspect shall be taken into account when processing the value of an attribute types by a StrongRevisionLabelString.

[constr_1563]{DRAFT} **Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category** ⌈ The AUTOSAR standard reserves the following values of attribute SoftwareClusterDesign.category and SoftwareCluster.category:

- ROOT_SOFTWARE_CLUSTER
- SUB_SOFTWARE_CLUSTER

⌋*()*

[TPS_MANI_01163]{DRAFT} **Impact of values of category on the semantics of SoftwareCluster** ⌈ A SoftwareCluster of category ROOT_SOFTWARE_CLUSTER may refer to other SoftwareClusters of category SUB_SOFTWARE_CLUSTER in the role subSoftwareCluster and thereby offer a way to further break down the creation of a SoftwareCluster. ⌋*(RS_MANI_00035)*

[TPS_MANI_01115]{DRAFT} **Specification of executable software within SoftwareCluster** ⌈ One of the most prominent contents of an uploadable software package is the reference to the executable software.

Within the definition of a SoftwareCluster, this reference is implicitly given by means of the reference SoftwareCluster.containedProcess.

The target of SoftwareCluster.containedProcess is a Process that represents an instance of the corresponding executable program (the software image), formalized as Executable ⌋*(RS_MANI_00035)*

The prominence of the dedicated reference to Process is amplified by the fact that it would have been technically possible to let Process inherit from UploadablePackageElement and thus include the referenced Process(es) in the bulk of references to other required model elements.

These references are formalized in two different forms. For technical reasons it is not possible to let all model elements that need to be immediately referenced by a SoftwareCluster inherit from UploadablePackageElement.

The main reason is that further model elements need to be referenced by a SoftwareCluster that are also used on the *AUTOSAR classic platform*.

In other words, it would be very questionable to introduce the “useless” concept of an UploadablePackageElement into the scope of the *AUTOSAR classic platform* as a mere (and unwanted) side effect of providing a definition of the SoftwareCluster on the *AUTOSAR classic platform*.

The scope of a single SoftwareCluster in terms of a relations to a Machine is that all software contained in one SoftwareCluster is supposed to be uploaded to one and only one Machine.

The definition of SoftwareCluster shall never include multiple Machines. This is expressed in [constr_1536].

[constr_1536]{DRAFT} **Definition of SoftwareCluster applies for a single Machine** ⌈ Within the scope of a SoftwareCluster, each Process referenced in the role containedProcess shall be mapped (e.g. by means of the existence of a ProcessToMachineMapping) to the same Machine. ⌋*()*

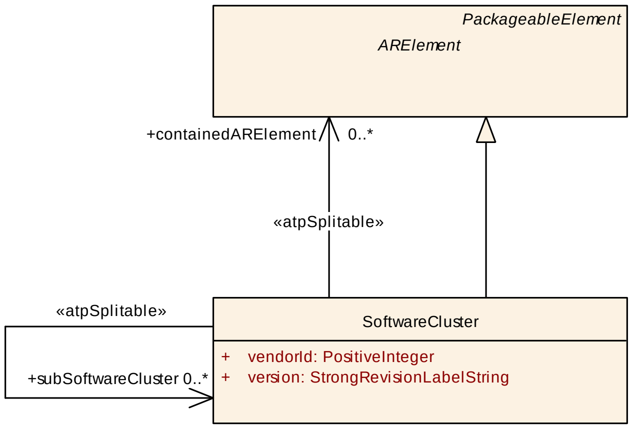


Figure 12.4: SoftwareCluster can reference ARElement

[TPS_MANI_01116]{DRAFT} **Reference to model elements included in an uploadable software package** ⌈ Beside the ability to explicitly reference a Process in the role containedProcess it is possible to define the following references to required model elements:

- references to meta-classes derived from UploadablePackageElement are formalized by way of SoftwareCluster.containedPackageElement.
- references to meta-classes derived from ARElement are formalized by way of SoftwareCluster.containedARElement.
- references to meta-classes derived from FibexElement are formalized by way of SoftwareCluster.containedFibexElement.

Technically, an UploadablePackageElement is also an ARElement, but it is still mandated to use the dedicated reference specifically for UploadablePackageElement. ⌋*(RS_MANI_00035)*

To exemplify the reference to UploadablePackageElement, Figure 12.3 contains a subclass of UploadablePackageElement: ServiceInstanceToPortPrototypeMapping.

It is obvious that the uploaded software needs to integrate with the communication stack and ServiceInstanceToPortPrototypeMapping is a prominent model element for this purpose.

[constr_1542]{DRAFT} **No nested definition of SoftwareCluster** ⌈ A SoftwareCluster shall not reference another SoftwareCluster in the role containedARElement. ⌋*()*

[TPS_MANI_01202]{DRAFT} **Semantics of reference SoftwareCluster.moduleInstantiation** ⌈ By means of the reference SoftwareCluster.moduleInstantiation it is possible to express the need for updates of the platform infrastructure along with other resources referenced by the enclosing SoftwareCluster. ⌋ *(RS_MANI_00035)*

[TPS_MANI_01218]{DRAFT} **Cryptographic signature of SoftwareCluster** ⌈ A SoftwareCluster also needs to be signed cryptographically. For this purpose, meta-class CryptoServiceCertificate is referenced in the role vendorSignature. ⌋ *(RS_MANI_00035)*

[TPS_MANI_01219]{DRAFT} **License of software in included SoftwareCluster** ⌈ It is possible to refer to licenses for software included in a SoftwareCluster by means of a reference to meta-class Documentation in the role license. ⌋ *(RS_MANI_00035)*

Class	**Documentation**			
Package	M2::AUTOSARTemplates::GenericStructure::DocumentationOnM1			
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 ARElement 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*			
Attribute	***Type***	***Mul.***	***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 12.5: Documentation

Please note that Documentation is an ARElement that cannot be owned by a SoftwareCluster. The latter can only refer to it.

This aspect also means that once a given license is formalized by means of a Documentation it is in general possible to refer to this formalization from within different SoftwareClusters.

[TPS_MANI_01220]{DRAFT} **Release notes of software in included SoftwareCluster** ⌈ It is possible to refer to release notes for software included in a SoftwareCluster by means of a reference to meta-class Documentation in the role releaseNotes. ⌋ *(RS_MANI_00035)*

12.3.2 Relevance of Software Cluster for Diagnostics

[TPS_MANI_01111]{DRAFT} **Diagnostic Address of a SoftwareCluster** ⌈ An uploadable software package formalized as a SoftwareCluster will typically be equipped with a diagnostics management component.

Therefore the definition of the SoftwareCluster needs to provide information about the diagnostic address(es) to which the contained diagnostic management component shall respond.

This information is formalized by means of the attribute SoftwareCluster.diagnosticAddress.

A SoftwareCluster may be required to respond to multiple (i.e. several functional plus one physical) diagnostic addresses, thus the multiplicity of diagnosticAddress is set to 0..*. ⌋*(RS_MANI_00035)*

Please note that the modeling of the SoftwareClusterDiagnosticAddress has been created with the primary goal to support the usage of DoIP for diagnostics.

The secondary goal has been to make the modeling of the diagnostic address extensible such that the idiomatic ways in which other transport layers (CAN, LIN, FlexRay, etc.) define diagnostic addresses can also be supported by adding respective subclasses of SoftwareClusterDiagnosticAddress.

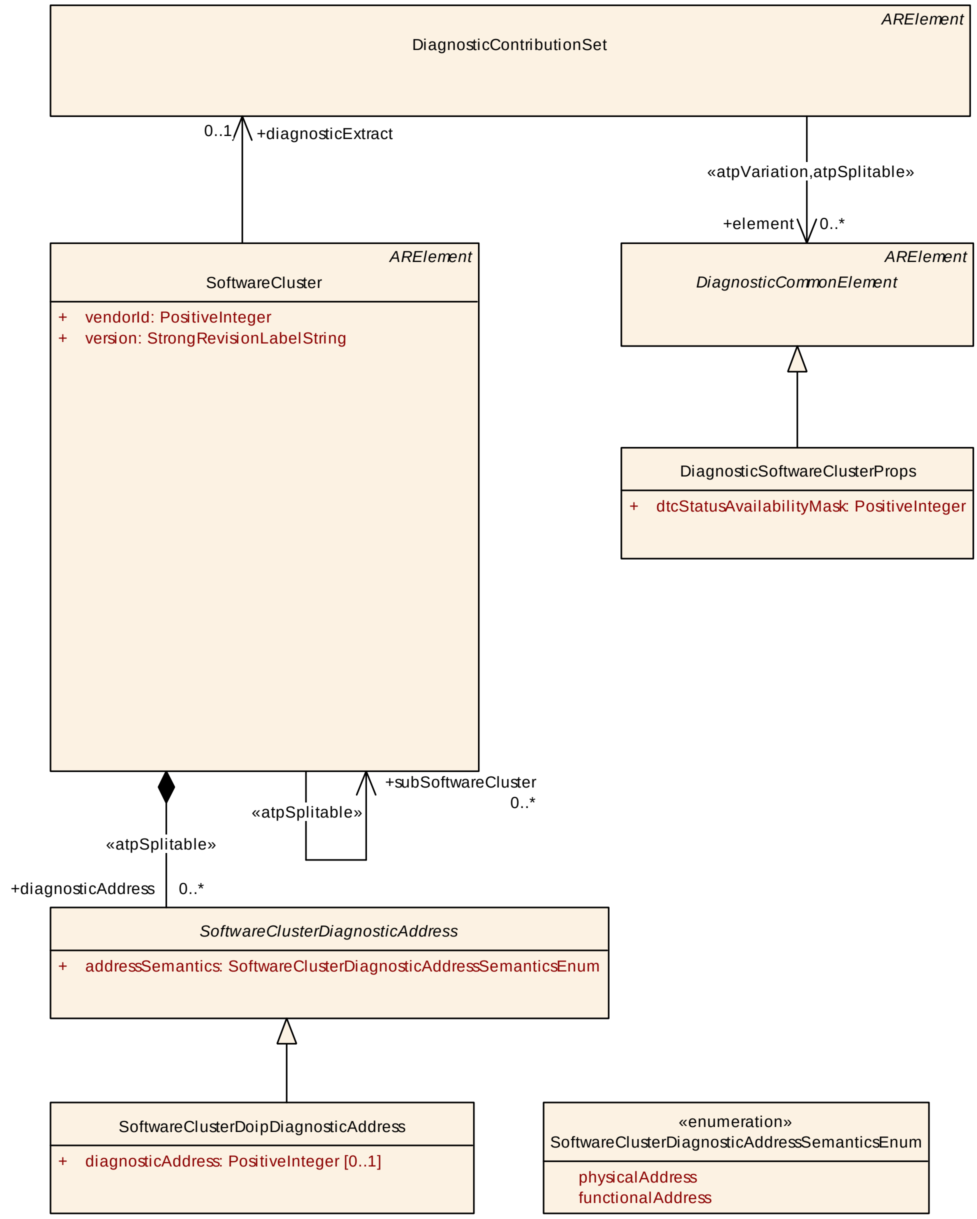


Figure 12.5: Modeling of diagnostic address of a SoftwareCluster

[constr_1543]{DRAFT} **Only one physical address per SoftwareCluster** ⌈ Each SoftwareCluster shall only aggregate one SoftwareClusterDiagnosticAddress where the value of attribute addressSemantics is set to SoftwareClusterDiagnosticAddressSemanticsEnum.physicalAddress. ⌋*()*

Class	**SoftwareCluster**
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage

▽

△

Class	SoftwareCluster			
Note	This meta-class represents the ability to define an uploadable software-package, i.e. the SoftwareCluster shall contain all software and configuration for a given purpose. **Tags:** atp.Status=draft atp.recommendedPackage=SoftwareClusters			
Base	*ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareCluster shall not be installed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=conflictsTo atp.Status=draft
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackage Element and that contribute to the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft
containedFibex Element	FibexElement	*	ref	This allows for referencing FibexElements that need to be considered in the context of a SoftwareCluster. **Tags:** atp.Status=draft
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedPackageElement atp.Status=draft
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster. **Tags:** atp.Status=draft
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dependsOn atp.Status=draft
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all Software ClusterDesigns applicable for the enclosing Software Cluster. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
diagnostic Address	SoftwareCluster DiagnosticAddress	*	aggr	This aggregation represents the collection of diagnostic addresses that apply for the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticAddress atp.Status=draft
diagnostic Extract	DiagnosticContribution Set	0..1	ref	This reference represents the definition of the diagnostic extract applicable to the referencing SoftwareCluster **Tags:** atp.Status=draft

▽

△

Class	SoftwareCluster			
license	Documentation	*	ref	This attribute allows for the inclusion of the the full text of a license of the enclosing SoftwareCluster. In many cases open source licenses require the inclusion of the full license text to any software that is released under the respective license. **Tags:** atp.Status=draft
module Instantiation	AdaptiveModule Instantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation atp.Status=draft
releaseNotes	Documentation	0..1	ref	This attribute allows for the explanations of changes since the previous version. The list of changes might require the creation of multiple paragraphs of test. **Tags:** atp.Status=draft
subSoftware Cluster	SoftwareCluster	*	ref	This reference is used to identify the sub-Software Clusters of an "umbrella" SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=subSoftwareCluster atp.Status=draft
vendorId	PositiveInteger	1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list.
vendor Signature	CryptoService Certificate	1	ref	This reference identifies the certificate that represents the vendor's signature. **Tags:** atp.Status=draft
version	StrongRevisionLabel String	1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

Table 12.6: SoftwareCluster

Class	***SoftwareClusterDiagnosticAddress*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability to define a diagnostic address in an abstract form. Sub-classes are supposed to clarify how the diagnostic address shall be defined according to the applicable addressing scheme (DoIP vs. CAN TP vs. ...). **Tags:** atp.Status=draft			
Base	*ARObject*			
Subclasses	SoftwareClusterDoipDiagnosticAddress			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
address Semantics	SoftwareCluster DiagnosticAddress SemanticsEnum	1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address.

Table 12.7: SoftwareClusterDiagnosticAddress

Enumeration	**SoftwareClusterDiagnosticAddressSemanticsEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage
Note	This meta-class defines a list of semantics for the interpretation of diagnostic addresses in the context of a SoftwareCluster. **Tags:** atp.Status=draft
Literal	***Description***
functionalAddress	This address represents a functional address. **Tags:** atp.EnumerationValue=1
physicalAddress	This address represents a physical address. **Tags:** atp.EnumerationValue=0

Table 12.8: SoftwareClusterDiagnosticAddressSemanticsEnum

Class	**SoftwareClusterDoipDiagnosticAddress**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability to define a diagnostic address specifically for the DoIP case. **Tags:** atp.Status=draft			
Base	*ARObject*, *SoftwareClusterDiagnosticAddress*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
diagnostic Address	PositiveInteger	0..1	attr	This attribute represents the collection of diagnostic addresses the SoftwareCluster occupies. **Tags:** atp.Status=draft

Table 12.9: SoftwareClusterDoipDiagnosticAddress

[TPS_MANI_01114]{DRAFT} **Relation of DiagnosticContributionSet to SoftwareCluster** ⌈ In AUTOSAR, the formalization of the external behavior of the diagnostic stack is rooted in meta-class DiagnosticContributionSet.

On the *AUTOSAR classic platform* the scope of the “external behavior of the diagnostic stack” is represented by an entire ECU.

This relation changes on the *AUTOSAR adaptive platform* where each uploadable software package is shipped with the definition of the “external behavior of the diagnostic stack” **as far as the software in the scope of respective uploadable software package is concerned**.

To fully support the different approaches of *AUTOSAR classic platform* and *AUTOSAR adaptive platform* it is necessary to provide means for specifying a DiagnosticContributionSet for a given SoftwareCluster.

In particular, this relation is created by means of the reference SoftwareCluster.diagnosticExtract. ⌋*(RS_MANI_00035)*

In other words, the “external behavior of the diagnostic stack” of each SoftwareCluster shall only be described by a single DiagnosticContributionSet.

And since the DiagnosticContributionSet and all referenced elements are subject to the upload on a target platform it only makes sense that the SoftwareCluster references the DiagnosticContributionSet (instead of the other way round).

[constr_1568]{DRAFT} **Existence of SoftwareCluster.diagnosticExtract** ⌈ The existence of the reference SoftwareCluster.diagnosticExtract is limited to SoftwareClusters where attribute category is set to the value ROOT_SOFTWARE_CLUSTER. ⌋*()*

Rationale for the existence of [constr_1562]: the definition of the diagnostic behavior is limited to the root level of a structure of SoftwareClusters in the same spirit that caused the existence of [constr_1564].

[constr_1534]{DRAFT} **Existence of DiagnosticSoftwareClusterProps** ⌈ Each DiagnosticContributionSet shall only reference one and only one DiagnosticSoftwareClusterProps in the role element. ⌋*()*

Class	**DiagnosticSoftwareClusterProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability to specify properties for the relation between a Diagnostic ContributionSet and a SoftwareCluster. **Tags:** atp.Status=draft atp.recommendedPackage=DiagnosticSoftwareClusterPropss			
Base	*ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dtcStatus AvailabilityMask	PositiveInteger	1	attr	This attribute contains the value of the DTC status availability mask. **Tags:** atp.Status=draft

Table 12.10: DiagnosticSoftwareClusterProps

[constr_1535]{DRAFT} **Existence of DiagnosticSoftwareClusterProps in the context of a DiagnosticContributionSet** ⌈ Each DiagnosticContributionSet shall only reference a single DiagnosticSoftwareClusterProps in the role element. ⌋*()*

[constr_1564]{DRAFT} **Existence of SoftwareCluster.diagnosticAddress** ⌈ The aggregation of SoftwareClusterDiagnosticAddress at SoftwareCluster in the role diagnosticAddress shall only exist if the value of SoftwareCluster.category is set to ROOT_SOFTWARE_CLUSTER. ⌋*()*

12.3.3 Sub Software Cluster

[constr_1565]{DRAFT} **Existence of SoftwareCluster.subSoftwareCluster** ⌈ The Reference from meta-class SoftwareCluster to itself in the role subSoftwareCluster shall only exist if the value of attribute SoftwareCluster.category is set to ROOT_SOFTWARE_CLUSTER. ⌋*()*

[constr_1566]{DRAFT} **Usage of SoftwareCluster.containedARElement** ⌈ The reference SoftwareCluster.containedARElement shall not be used to refer to another SoftwareCluster or even SoftwareCluster. ⌋*()*

Rationale for the existence of [constr_1566]: dedicated references are defined for the purpose of referring to SoftwareClusters.

12.3.4 Software Cluster Dependency

[TPS_MANI_01164]{DRAFT} **Semantics of SoftwareCluster.dependsOn** ⌈ A SoftwareCluster has the ability to express a dependency to other SoftwareClusters in the role dependsOn. The semantics is to express that the functionality of the referenced SoftwareCluster is required to enable the full functionality of the referencing SoftwareCluster.

Attribute SoftwareCluster.dependsOn allows for the definition of a **formal** (potentially nested) dependency condition. The dependency shall be applicable only if the condition defined by dependsOn yields True. ⌋*(RS_MANI_00035)*

[TPS_MANI_01214]{DRAFT} **Semantics of SoftwareCluster.conflictsTo** ⌈ A SoftwareCluster has the ability to express a conflict to other SoftwareClusters in the role conflictsTo. The semantics is to express that the functionality of the referenced SoftwareCluster **inhibits** the installation of the referencing SoftwareCluster.

Attribute SoftwareCluster.conflictsTo allows for the definition of a **formal** (potentially nested) dependency condition. The dependency shall be applicable only if the condition defined by conflictsTo yields False. ⌋*(RS_MANI_00035)*

Both the reference in the role SoftwareCluster.dependsOn/conflictsTo as well as the reference in the role SoftwareCluster.subSoftwareCluster factually define a dependency between SoftwareClusters.

However, the difference is that SoftwareClusters referenced in the role subSoftwareCluster is not yet installed while SoftwareCluster.dependsOn/conflictsTo refers to a SoftwareCluster that is already installed on the target platform.

A formal difference between the two is that the SoftwareCluster referenced in the role subSoftwareCluster may not have subSoftwareClusters themselves (as explained by [constr_1559]).

That said, it could not really happen that SoftwareCluster.dependsOn/conflictsTo finally refers to a SoftwareCluster that is also referenced in the role subSoftwareCluster. Nevertheless, [constr_1567] regulates the usage of SoftwareCluster.dependsOn/conflictsTo vs. subSoftwareCluster.

[constr_1567]{DRAFT} **Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn/conflictsTo** ⌈ Within the context of a specific SoftwareCluster, the references from SoftwareCluster to itself in the roles subSoftwareCluster and dependsOn/conflictsTo shall not refer to the same target SoftwareCluster. ⌋()

Rationale for the existence of [constr_1567]: the existence of a reference to another SoftwareCluster in two different roles creates ambiguity and also there is no definition for the semantics of such a scenario.

[TPS_MANI_01215]{DRAFT} **Semantics of meta-class SoftwareClusterDependencyFormula** ⌈ Meta-class SoftwareClusterDependencyFormula allows for the definition of a formal condition that can be taken to decide about the dependency to or the conflict with a SoftwareCluster.

The modeling of SoftwareClusterDependencyFormula allows for the definition of nested conditions. The attribute operator is applied on the results of the evaluation of the parts. ⌋*(RS_MANI_00035)*

Class	**SoftwareClusterDependencyFormula**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability to define a dependency among SoftwareClusters. **Tags:** atp.Status=draft			
Base	*ARObject*, *SoftwareClusterDependencyFormulaPart*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
operator	SoftwareClusterDependencyLogicalOperatorEnum	0..1	attr	This logical operator can be used to relate the results of different SoftwareClusterDependencyParts.
part (ordered)	SoftwareClusterDependencyFormulaPart	*	aggr	This aggregation represents the ordered collection of the parts of the SoftwareClusterDependencyFormula. **Tags:** atp.Status=draft

Table 12.11: SoftwareClusterDependencyFormula

[TPS_MANI_01216]{DRAFT} **Semantics of meta-class SoftwareClusterDependencyFormulaPart** ⌈ Meta-class SoftwareClusterDependencyFormulaPart represents a part of a SoftwareClusterDependencyFormula. The order of the parts of a SoftwareClusterDependencyFormula is significant. ⌋*(RS_MANI_00035)*

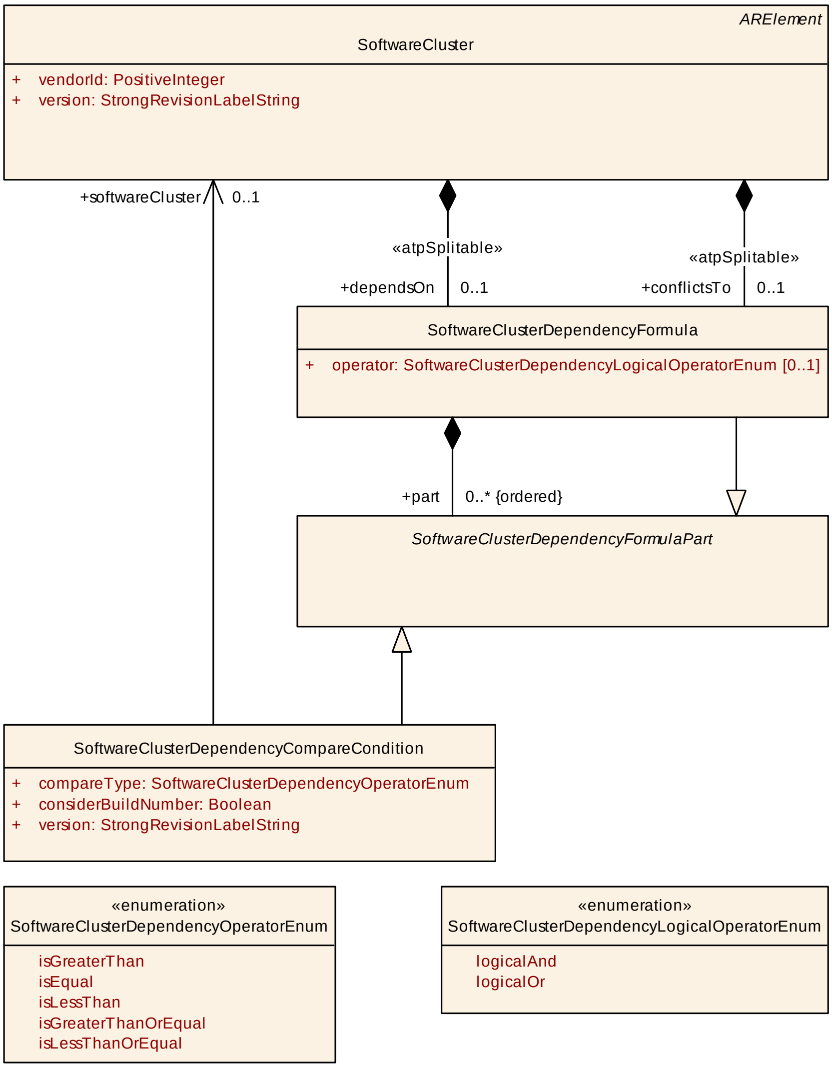


Figure 12.6: Modeling of dependencies in the context of a SoftwareCluster

At the same time, SoftwareClusterDependencyFormulaPart is the base class of SoftwareClusterDependencyFormula. This means that the SoftwareClusterDependencyFormula can aggregate all subclasses of SoftwareClusterDependencyFormulaPart, i.e. SoftwareClusterDependencyFormula and SoftwareClusterDependencyCompareCondition.

Class	***SoftwareClusterDependencyFormulaPart*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage
Note	This meta-class represents an abstract base class for the definition of different formula parts of a SoftwareClusterDependencyFormula. **Tags:** atp.Status=draft
Base	*ARObject*
Subclasses	SoftwareClusterDependencyCompareCondition, SoftwareClusterDependencyFormula

△

Class	***SoftwareClusterDependencyFormulaPart*** (abstract)			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 12.12: SoftwareClusterDependencyFormulaPart

Class	**SoftwareClusterDependencyCompareCondition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability to specify a concrete dependency condition in the context of a SoftwareClusterDependencyFormula. **Tags:** atp.Status=draft			
Base	*ARObject*, *SoftwareClusterDependencyFormulaPart*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
compareType	SoftwareCluster DependencyOperator Enum	1	attr	This attribute identifies the semantics of the compare operator.
considerBuild Number	Boolean	1	attr	If this attribute is set to true then the build number shall be taken into account for the comparison. Build numbers don't have to be consecutive but could be created by some kind of hashing algorithm. In such a case it might make sense to include the build number in a test for equality but it is probably not reasonable to apply e.g. a less-than comparison.
softwareCluster	SoftwareCluster	0..1	ref	This reference identifies the SoftwareCluster to which the dependency/confict applies. **Tags:** atp.Status=draft
version	StrongRevisionLabel String	1	attr	This attribute represents the value of a version against which the comparison shall be executed.

Table 12.13: SoftwareClusterDependencyCompareCondition

This modeling pattern obviously allows for the creation of nested trees of dependencies (or conflicts) from the point of view of a given `SoftwareCluster`.

Figure 12.7 provides a conceptual sketch of the modeling of dependencies among `SoftwareCluster`s.

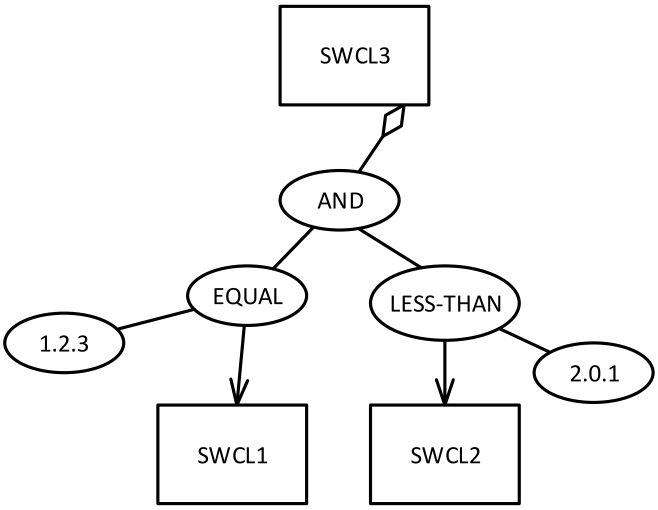


Figure 12.7: Tree of dependency conditions

[TPS_MANI_01217]{DRAFT} **Semantics of metaclass SoftwareClusterDependencyCompareCondition** ⌈ Meta-class SoftwareClusterDependencyCompareCondition allows for the definition of a formal condition to compare against the version of the referenced softwareCluster using a given compareType.

The ability to specifically decide about whether or not to consider the build number in the comparison is implemented by means of attribute considerBuildNumber. ⌋ *(RS_MANI_00035)*

12.4 Software Package

The existence of the SoftwareCluster by itself is not sufficient for installation. Actually, the SoftwareCluster gets wrapped into a so-called SoftwarePackage that comes with an own manifest format that is at least partly standardized.

The difference between the semantics of a SoftwareCluster and the semantics of SoftwarePackage is that a SoftwareCluster focusses on the structure of the software itself while the SoftwarePackage is created to handle the logistics aspect of the software installation.

[TPS_MANI_01221]{DRAFT} **Semantics of meta-class SoftwarePackage** ⌈ The purpose of meta-class SoftwarePackage is to cover the "logistics" aspect of the software installation procedure. ⌋ *(RS_MANI_00035)*

Class	SoftwarePackage			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability to formalize the content of a software package. **Tags:** atp.Status=draft atp.recommendedPackage=SoftwarePackages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
actionType	SoftwarePackageAction TypeEnum	1	attr	This attribute defines the action to be taken in the step of processing the enclosing SoftwarePackage.
activationAction	SoftwarePackage ActivationActionEnum	1	attr	This attribute governs the action to be taken after the installation of the SoftwareCluster completed.
compressed Software PackageSize	PositiveInteger	1	attr	This size represents the size of the compressed Software Package.
isDeltaPackage	Boolean	1	attr	This attribute denotes whether the SoftwarePackage is only able to update but not for initial installation.
maximum SupportedUcm Version	RevisionLabelString	1	attr	This attribute identifies the maximum supported version of the UCM for this SoftwarePackage.
minimum SupportedUcm Version	RevisionLabelString	1	attr	This attribute identifies the minimum supported version of the UCM for this SoftwarePackage.
packagerId	PositiveInteger	1	attr	This attribute identifies Id of the organization that provides the packager generating the SoftwarePackage.
packager Signature	CryptoService Certificate	1	ref	This reference identifies the certificate that represents the packager's signature. **Tags:** atp.Status=draft
softwareCluster	SoftwareCluster	1	ref	This reference identifies the SoftwareCluster that belongs to the SoftwarePackage. The nature of this relation is actually more like an aggregation than a reference. But the relation is still modelled as a reference because two ARElements cannot aggregate each other. **Tags:** atp.Status=draft
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country.
uncompressed SoftwareCluster Size	PositiveInteger	1	attr	This attribute gives an indication about the storage that has to be available on the target.

Table 12.14: SoftwarePackage

[constr_1690]{DRAFT} **SoftwareCluster shall only be referenced by a single SoftwarePackage.** ⌈ Each SoftwareCluster shall only be referenced by a single SoftwarePackage. ⌋*()*

In other words, AUTOSAR factually assumes a 1:1 relation between SoftwarePackage and SoftwareCluster. Such a relation would otherwise typically be modeled by means of an aggregation with the multiplicity 1.

However, a SoftwareCluster is derived from base class PackageableElement which is only aggregated by ARPackage. Subclasses of PackageableElement - by convention - shall not be aggregated by any other meta-class.

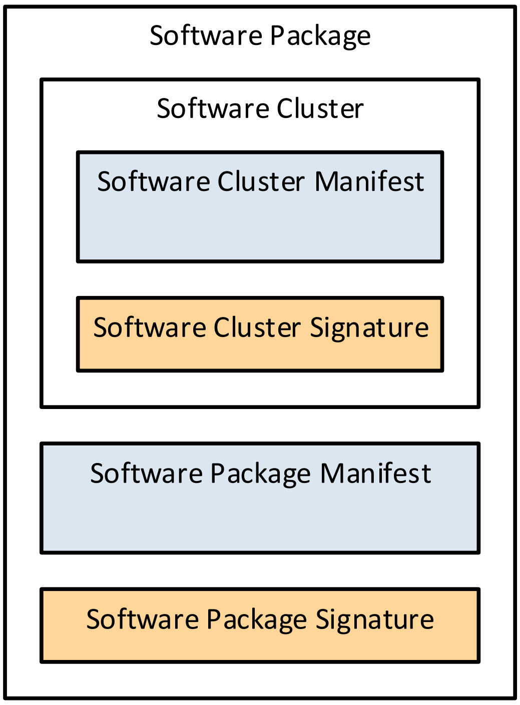


Figure 12.8: Conceptual relation of SoftwarePackage and SoftwareCluster

[TPS_MANI_01222]{DRAFT} **Cryptographic signature of SoftwarePackage** ⌈ A SoftwarePackage also needs to be signed cryptographically. For this purpose, meta-class CryptoServiceCertificate is referenced in the role packagerSignature. ⌋*(RS_MANI_00035)*

[TPS_MANI_01223]{DRAFT} **Semantics of attribute SoftwarePackage.packagerId** ⌈ Attribute SoftwarePackage.packagerId contains the value of the AUTOSAR vendor Id of the organization that created software tool that created the SoftwarePackage. ⌋*(RS_MANI_00035)*

For clarification, a UCM can only accept packages that are generated by a packaging tool developed by the same organization that also developed the UCM itself. The vendor of the SoftwareCluster contained in the SoftwarePackage can obviously be different.

[TPS_MANI_01224]{DRAFT} **Actions taken after installation of a SoftwarePackage** ⌈ After a SoftwarePackage has been installed on the target machine it is possible to execute one of the following actions:

- Reboot the target platform.
- Restart the installed SoftwareCluster.
- Do nothing.

These options are formalized by means of meta-class SoftwarePackageActivationActionEnum and attribute SoftwarePackage.activationAction. ⌋ (RS_MANI_00035)

In contrast to the action taken **after** the installation, it is also important to be able to define the semantics of the actual installation procedure.

Enumeration	**SoftwarePackageActivationActionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage
Note	This enumeration provides a choice of possible actions to be executed on installing a Software Package to a target Machine. **Tags:** atp.Status=draft
Literal	***Description***
nothing	The installation has no further consequences in terms of other software on the target. **Tags:** atp.EnumerationValue=2
reboot	Reboot the whole Machine. **Tags:** atp.EnumerationValue=0
restartApplication	Restart the application software on the target Machine. **Tags:** atp.EnumerationValue=1

Table 12.15: SoftwarePackageActivationActionEnum

[TPS_MANI_01225]{DRAFT} **Actions taken during installation of a SoftwarePackage** ⌈ It is necessary to define the concrete activity that shall be taken to handle the SoftwarePackage on the target machine. Possible actions are:

- Do a clean installation of a SoftwareCluster.
- Update a previously installed SoftwareCluster.
- Remove a SoftwareCluster

These options are formalized by means of meta-class SoftwarePackageActionTypeEnum and attribute SoftwarePackage.actionType. ⌋ (RS_MANI_00035)

Enumeration	**SoftwarePackageActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage
Note	**Tags:** atp.Status=draft
Literal	***Description***
install	**Tags:** atp.EnumerationValue=1
remove	**Tags:** atp.EnumerationValue=2
update	**Tags:** atp.EnumerationValue=0

Table 12.16: SoftwarePackageActionTypeEnum

A Examples

This chapter contains a collection of examples that reflect concepts described in different chapters of this document. The content of the chapter provides mere explanation and does not add anything to the model semantics.

A.1 Service Instance Deployment by Service Interface Mapping

The example in Figure A.2 sketches the modeling of a ProvidedSomeipServiceInstance in the presence of a ServiceInterfaceMapping, that references two ServiceInterfaces in the role sourceServiceInterface.

For support, Figure A.1 contains an excerpt from the meta-model that contains the relevant meta-classes that have been instantiated to create the example sketched in Figure A.2.

Note further that the example depicted in Figure A.2 is not limited to the explanation of the actual ServiceInterfaceMapping.

As the main use case for this is the usage of ServiceInterfaces for the definition of an “outside” communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

Please note that the modeling of the binding requires the existence of a PortPrototype, which in turn is aggregated by an SwComponentType (not depicted).

This approach still contains some degrees of freedom with respect to the role of the SwComponentType that aggregates the mentioned PortPrototype. This document does not go further in discussing the nature of such a configuration.

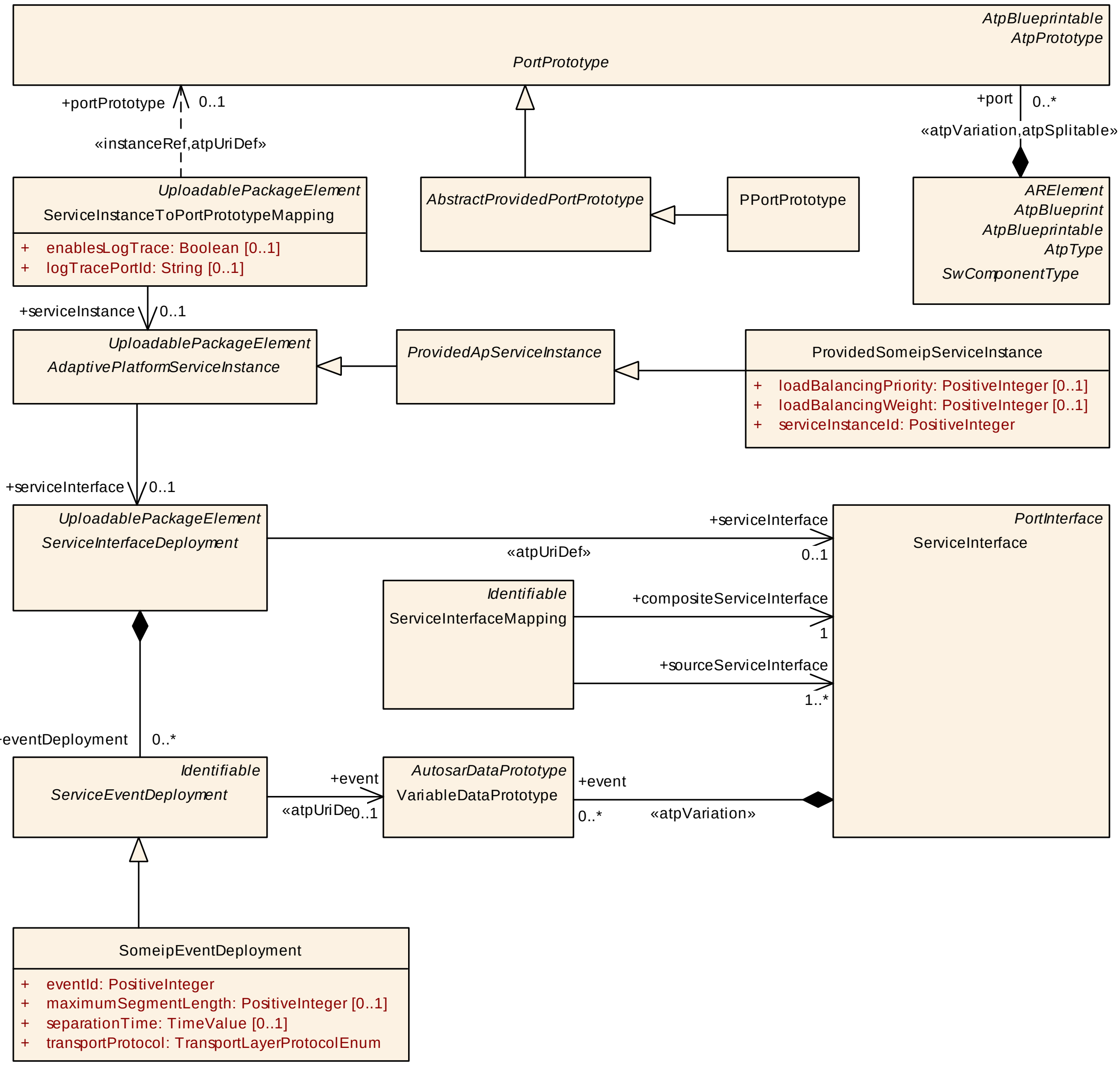


Figure A.1: Meta-model excerpt relevant for the example

For reasons of keeping the example as simple as possible, each of the ServiceInterfaces in the role sourceServiceInterface aggregate a single event.

The ServiceInterface referenced in the role compositeServiceInterface aggregates two event with shortNames that match the mentioned event of the source ServiceInterfaces (see [TPS_MANI_01022]).

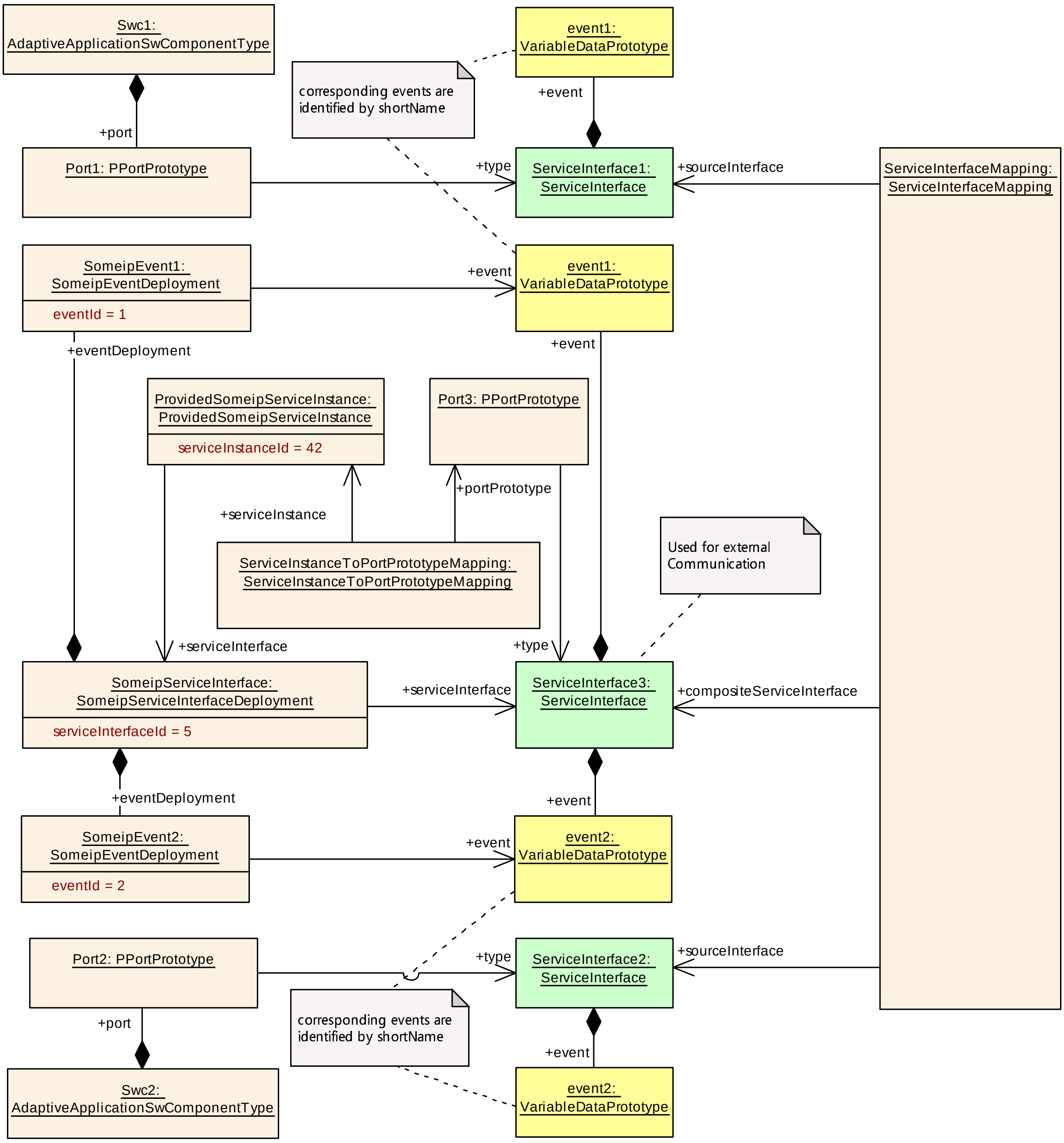


Figure A.2: Example for the deployments of a service in the presence of a `ServiceInterfaceMapping`

A.2 Service Instance Deployment by Service Interface Element Mapping

The example in Figure A.4 sketches the modeling of a `ProvidedSomeipServiceInstance` in the presence of a `ServiceInterfaceEventMapping`s. In principle, this example is very close to the example described in Figure A.2.

In contrast to the example sketched in Figure A.2, the example depicted in Figure A.4 uses a mapping to individual elements of a ServiceInterface instead of the entire ServiceInterface.

Please find the corresponding excerpt of relevant meta-classes for the utilization of ServiceInterfaceEventMapping sketched in Figure A.3.

Note further that the example depicted in Figure A.3 is not limited to the explanation of the actual ServiceInterfaceElementMapping.

As the main use case for this is the usage of ServiceInterfaces for the definition of an "outside" communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

Please note that the modeling of the binding requires the existence of a PortPrototype, which in turn is aggregated by an SwComponentType (not depicted).

This approach still contains some degrees of freedom with respect to the role of the SwComponentType that aggregates the mentioned PortPrototype. This document does not go further in discussing the nature of such a configuration.

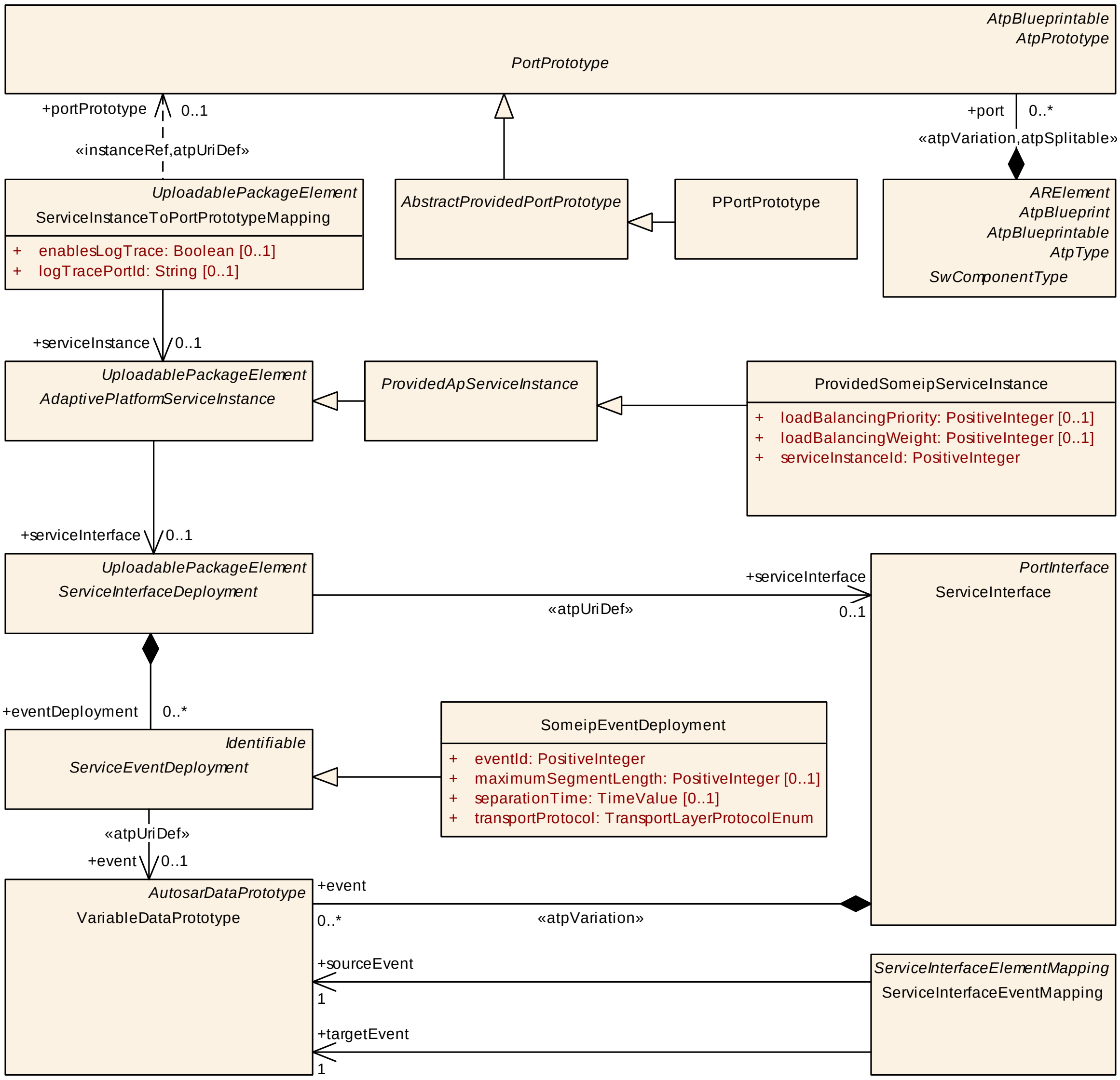


Figure A.3: Excerpt of the relevant meta-classes for the ServiceInterfaceEventMapping example

By mapping individual elements of ServiceInterfaces, it is possible to map element with different shortNames to each other. In this example, the event with the shortName event1 is mapped to another event with the shortName eventLeft.

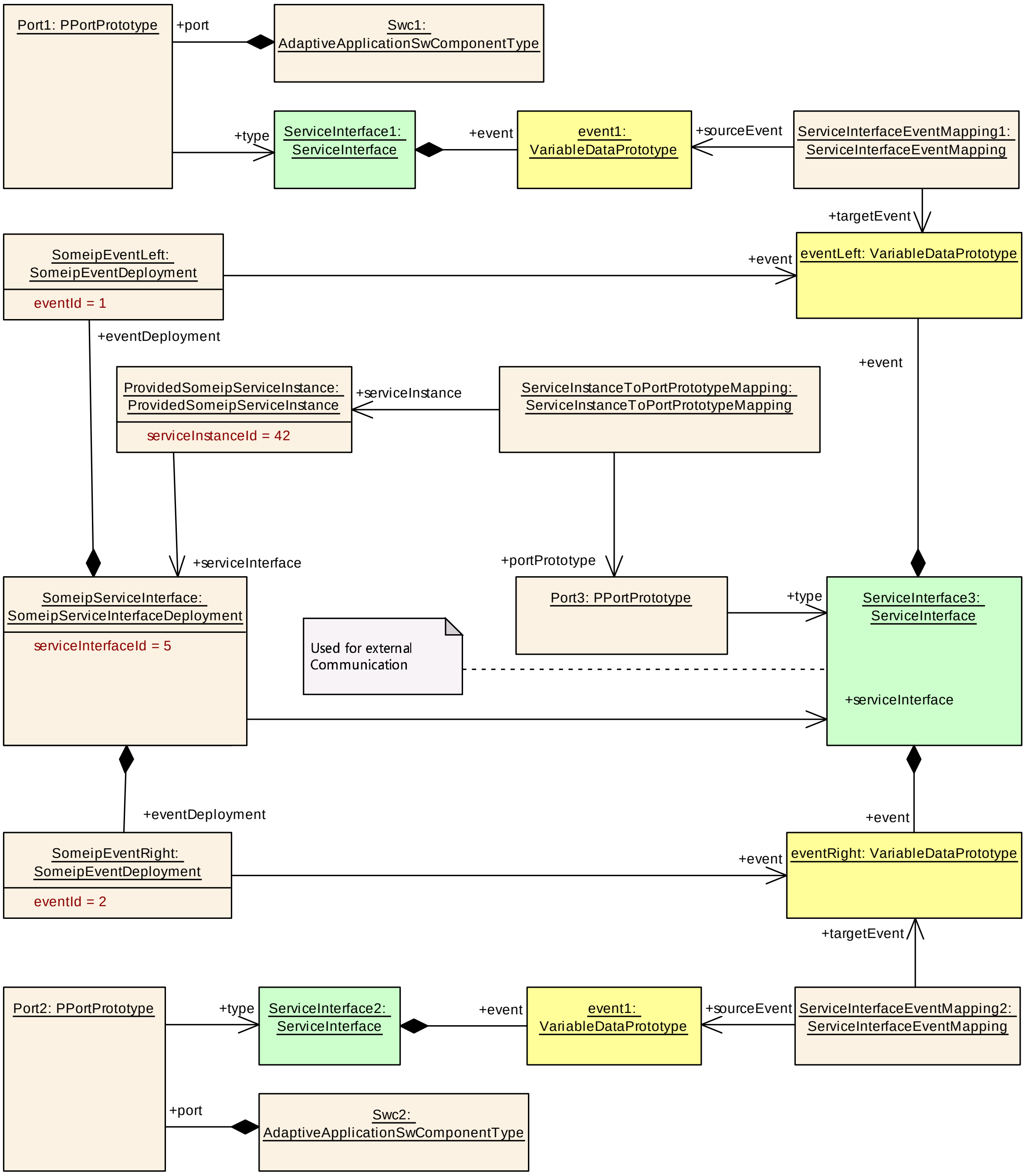


Figure A.4: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping`

In Figure A.4, two different `ServiceInterface`s exist that each aggregate an `event` with the identical `shortName`. This scenario **requires** the existence of `ServiceInterfaceElementMapping`s.

As an extension to the scenario depicted in Figure A.4, Figure A.5 describes a model where the **same** `event` of a `ServiceInterface` is used in two different event deployments by means of two `ServiceInterfaceEventMapping`s that each refer to said `event` in the role `ServiceInterfaceEventMapping.sourceEvent`.

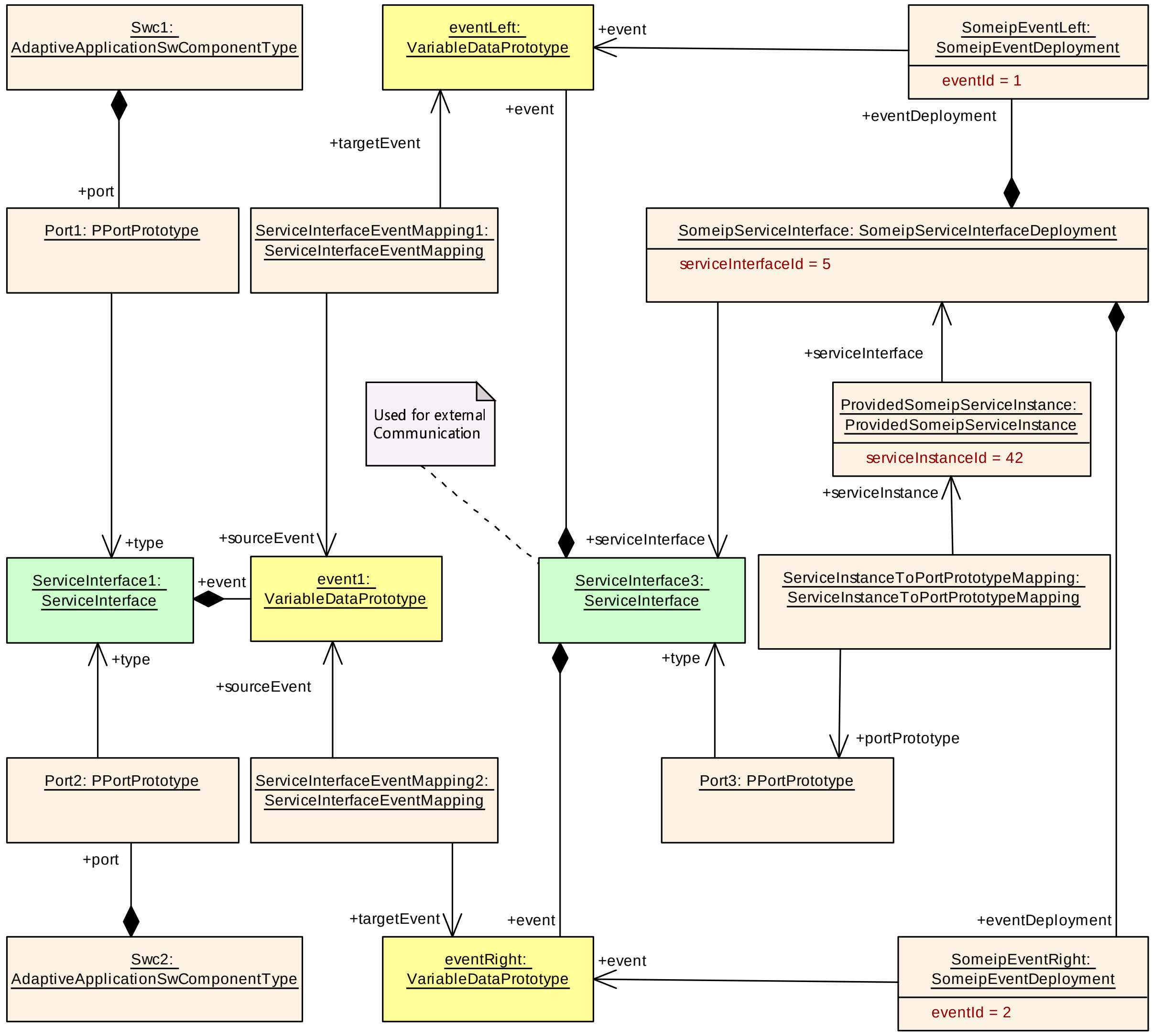


Figure A.5: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping` to the same source `ServiceInterface`

Again, this scenario **requires** the existence of appropriately configured `ServiceInterfaceElementMapping`s.

A.3 Definition of Startup Configuration

As already mentioned, the mode-dependent startup configuration is directly aggregated by the definition of a `Process`:

```
<PROCESS>
  <SHORT-NAME>AA1</SHORT-NAME>
  <STATE-DEPENDENT-STARTUP-CONFIGS>
    <STATE-DEPENDENT-STARTUP-CONFIG>
      <EXECUTION-DEPENDENCYS>
        <EXECUTION-DEPENDENCY>
          <PROCESS-STATE-IREF>
```

```
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
DECLARATION-GROUP-PROTOTYPE">/Processes/MWC/ProcessStateMachine</CONTEXT
-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
ModeDeclarationGroups/ProcessStateMachine/Running</TARGET-MODE-
DECLARATION-REF>
        </PROCESS-STATE-IREF>
      </EXECUTION-DEPENDENCY>
      <EXECUTION-DEPENDENCY>
        <PROCESS-STATE-IREF>
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
DECLARATION-GROUP-PROTOTYPE">/Processes/MSM/ProcessStateMachine</CONTEXT
-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
ModeDeclarationGroups/ProcessStateMachine/Running</TARGET-MODE-
DECLARATION-REF>
        </PROCESS-STATE-IREF>
      </EXECUTION-DEPENDENCY>
    </EXECUTION-DEPENDENCYS>
    <FUNCTION-GROUP-STATE-IREFS>
      <FUNCTION-GROUP-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
DECLARATION-GROUP-PROTOTYPE">/Machines/ExampleMachine/
ExampleMachine_StateMachine</CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-
REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
ModeDeclarationGroups/VehicleStateMachine/Driving</TARGET-MODE-
DECLARATION-REF>
      </FUNCTION-GROUP-STATE-IREF>
    </FUNCTION-GROUP-STATE-IREFS>
    <RESOURCE-GROUP-REF DEST="RESOURCE-GROUP">/Machines/ExampleMachine/
Linux/resourceGroup2</RESOURCE-GROUP-REF>
    <STARTUP-CONFIG-REF DEST="STARTUP-CONFIG">/StartupConfigSets/
StartupConfigSet_AA/AA1_Startup</STARTUP-CONFIG-REF>
  </STATE-DEPENDENT-STARTUP-CONFIG>
 </STATE-DEPENDENT-STARTUP-CONFIGS>
</PROCESS>
```

Listing A.1: Example for the definition of the StateDependentStartupConfig owned by a Process

In this example, launch dependencies exist on two other Processes. Both Processes MWC and MSM need to be in the ProcessState “Running” before AA1 is started.

The reference StateDependentStartupConfig.functionGroupState refers to a ModeDeclaration with the shortName Driving within the state machine of the underlying Machine. In other words the referenced StartupConfig that is defined in Listing A.2 is valid if the Machine is in the machine state Driving.

The StateDependentStartupConfig of the Process is assigned to the ResourceGroup named ResourceGroup2 that is defined in the Machine Manifest.

```
<STARTUP-CONFIG>
  <SHORT-NAME>AA1_Startup</SHORT-NAME>
  <SCHEDULING-POLICY>SCHEDULING-POLICY-FIFO</SCHEDULING-POLICY>
  <SCHEDULING-PRIORITY>20</SCHEDULING-PRIORITY>
```

```
  <STARTUP-OPTIONS>
    <STARTUP-OPTION>
      <OPTION-ARGUMENT>inputfile_1</OPTION-ARGUMENT>
      <OPTION-KIND>COMMAND-LINE-LONG-FORM</OPTION-KIND>
      <OPTION-NAME>filename</OPTION-NAME>
    </STARTUP-OPTION>
  </STARTUP-OPTIONS>
</STARTUP-CONFIG>
```

Listing A.2: Example for a `StartupConfig`

Please note that the definition of the `StartupOption` in the example yields an actual command-line option that reads `--filename=inputfile_1`.

The corresponding definition of a `Machine` contains a `OsModuleInstantiation` that in turn owns the two `ResourceGroup`s named `ResourceGroup1` and `ResourceGroup2`. This aspect can be found in Listing A.3.

```
<MACHINE>
  <SHORT-NAME>ExampleMachine</SHORT-NAME>
  <FUNCTION-GROUPS>
    <MODE-DECLARATION-GROUP-PROTOTYPE>
      <SHORT-NAME>ExampleMachine_StateMachine</SHORT-NAME>
      <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/
   VehicleStateMachine</TYPE-TREF>
    </MODE-DECLARATION-GROUP-PROTOTYPE>
  </FUNCTION-GROUPS>
  <MODULE-INSTANTIATIONS>
    <OS-MODULE-INSTANTIATION>
      <SHORT-NAME>Linux</SHORT-NAME>
      <RESOURCE-GROUPS>
        <RESOURCE-GROUP>
          <SHORT-NAME>resourceGroup1</SHORT-NAME>
          <CPU-USAGE>60</CPU-USAGE>
          <MEM-USAGE>1000000</MEM-USAGE>
        </RESOURCE-GROUP>
        <RESOURCE-GROUP>
          <SHORT-NAME>resourceGroup2</SHORT-NAME>
          <CPU-USAGE>70</CPU-USAGE>
          <MEM-USAGE>2000000</MEM-USAGE>
        </RESOURCE-GROUP>
      </RESOURCE-GROUPS>
    </OS-MODULE-INSTANTIATION>
  </MODULE-INSTANTIATIONS>
</MACHINE>
```

Listing A.3: Example for the definition of a `Machine`

A.4 Service Instance Mapping

This section contains some examples that explain the modeling of a mapping between a service instance and the application. The examples have been created to show both the “find” and the “offer” side of the service binding.

In the first example, depicted in Figure A.6 shows the binding of PortPrototypes to a SOME/IP-based transport layer. The left part of the diagram contains the modeling of the "find" aspect and the right part contains the modeling of the "offer" aspect.

Please note that the shortNames of the two affected PortPrototypes are different. In other words, the shortNames of the PortPrototypes are not used as a way to identify the opposite end of the service binding.

Instead, the existence of a ServiceInstanceToPortPrototypeMapping that maps a PortPrototype to a ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance with the **identical value** of attribute serviceInstanceId creates the actual binding between the "find" and the "offer" end.

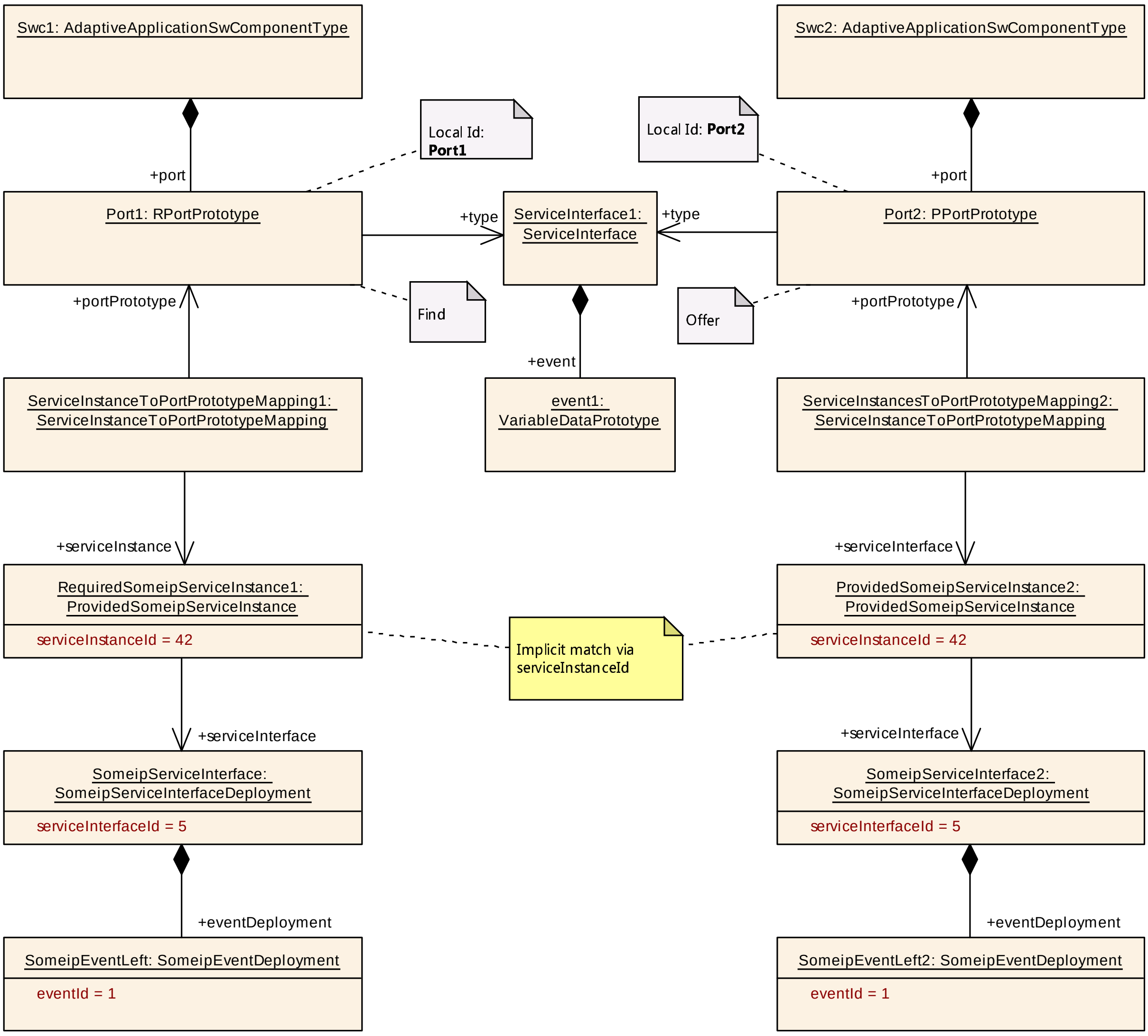


Figure A.6: Port-based binding of a service instance to the application using SOME/IP

The next example (depicted in Figure A.7) shows a binding of PortPrototypes to a user-defined transport layer. The left part of the diagram contains the modeling of the "find" aspect and the right part contains the modeling of the "offer" aspect.

Because the binding is user-defined, there are no attributes modeled on the level of the meta-model available to identify an instance according to the user-defined service implementation. There is just no way to define attributes that are "needed anyway" for a user-defined binding.

Therefore, the only option in this case it the usage of `AdminData`, `Sdg`, and `Sd` to define an identification of the user-defined transport layer.

In order to support the comparison to the example depicted in Figure A.6, the example described in Figure A.7 uses a simple identification based on a numerical value. Again, this is an arbitrary scenario created just for the sake of explanation.

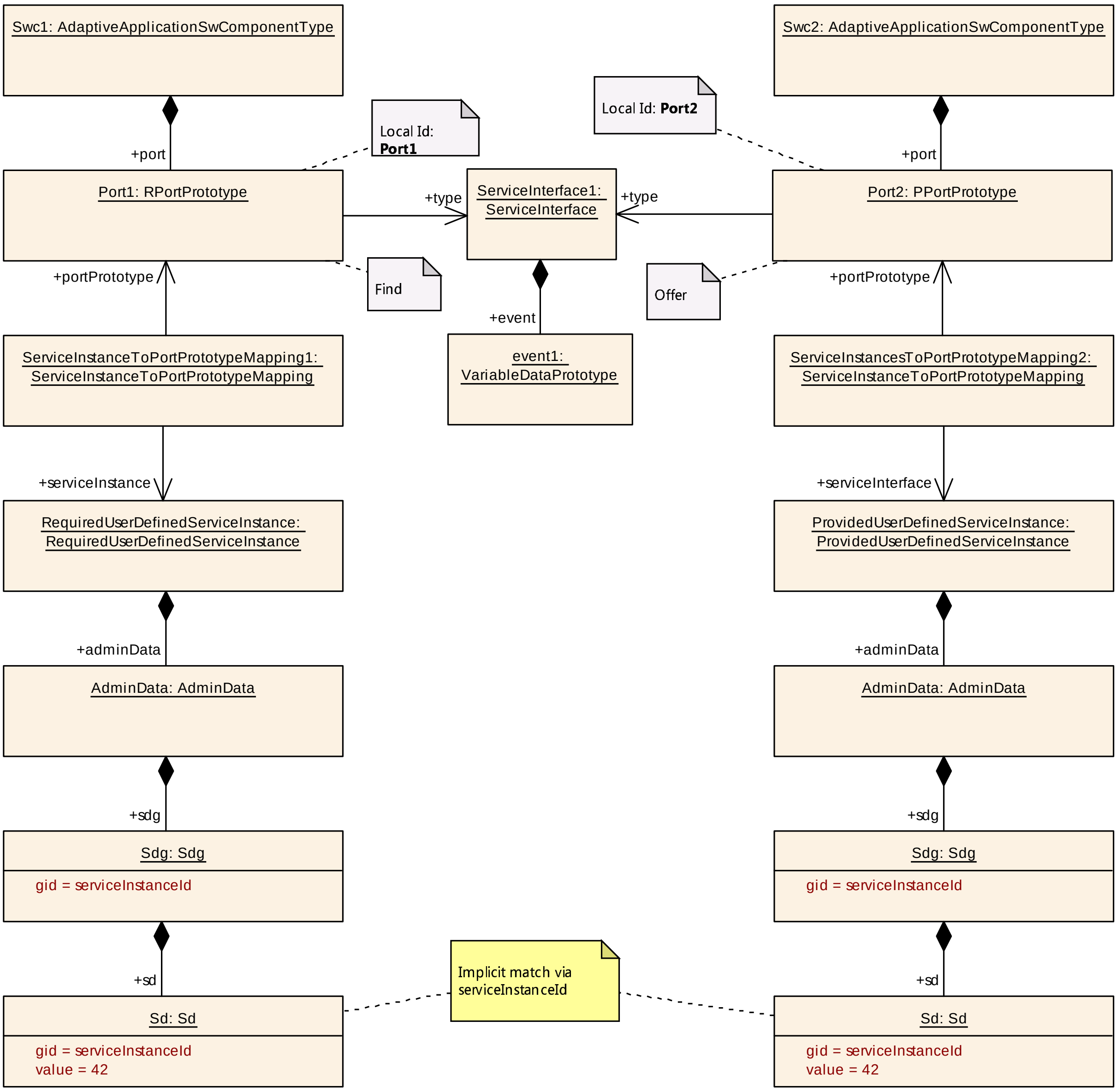


Figure A.7: Port-based binding of a service instance to the application using a user-defined binding

A.5 Radar and Camera ServiceInterface example

The example in figure A.8 shows a *Radar* `ServiceInterface` with a *BrakeEvent* and two `method`s: *Calibrate* and *Adjust*. The *Camera* `ServiceInterface` shown in figure A.9 has two events: *LaneEvent* and *SpeedLimitEvent* and one *Calibrate* `method`.

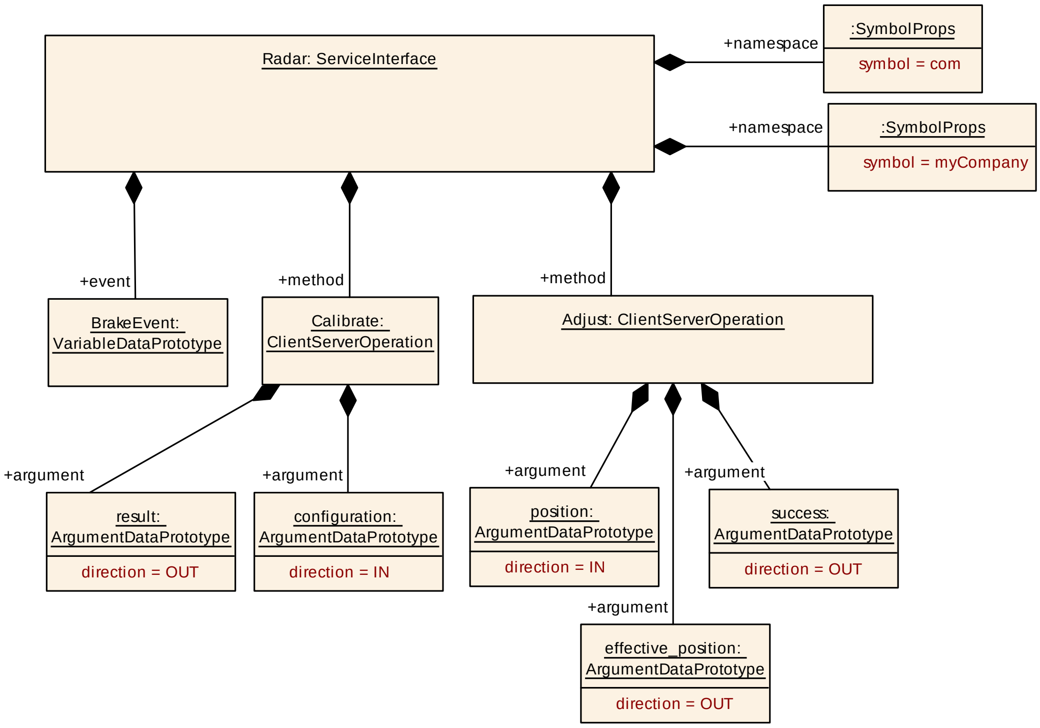


Figure A.8: Radar Service Interface

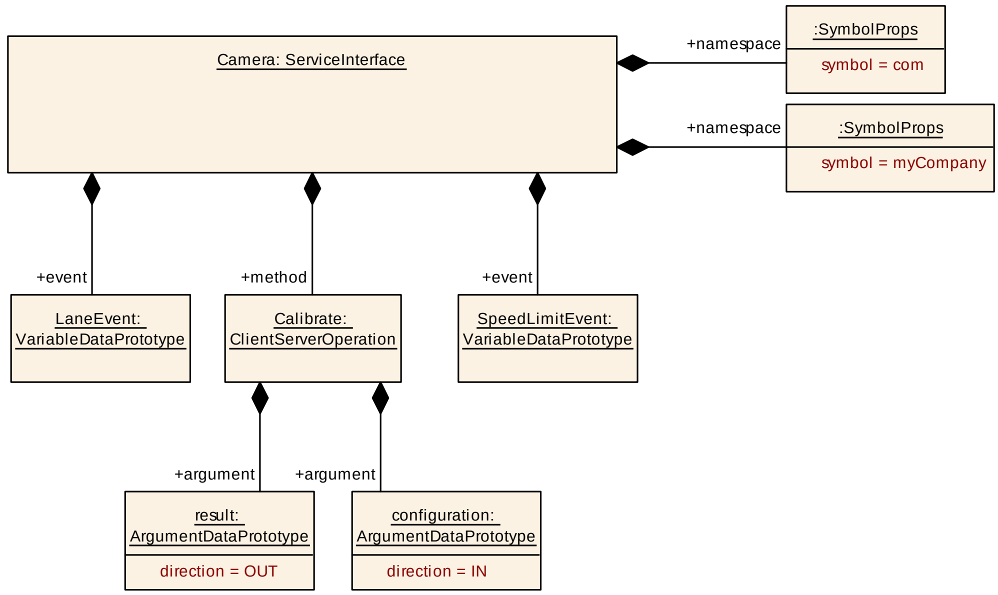


Figure A.9: Camera Service Interface

Both `ServiceInterface`s *Radar* and *Camera* are mapped to a combined *RadarAndCamera* `ServiceInterface` with an Service Interface Element Mapping since both `ServiceInterface`s have a `method` with the same name: *Calibrate*.

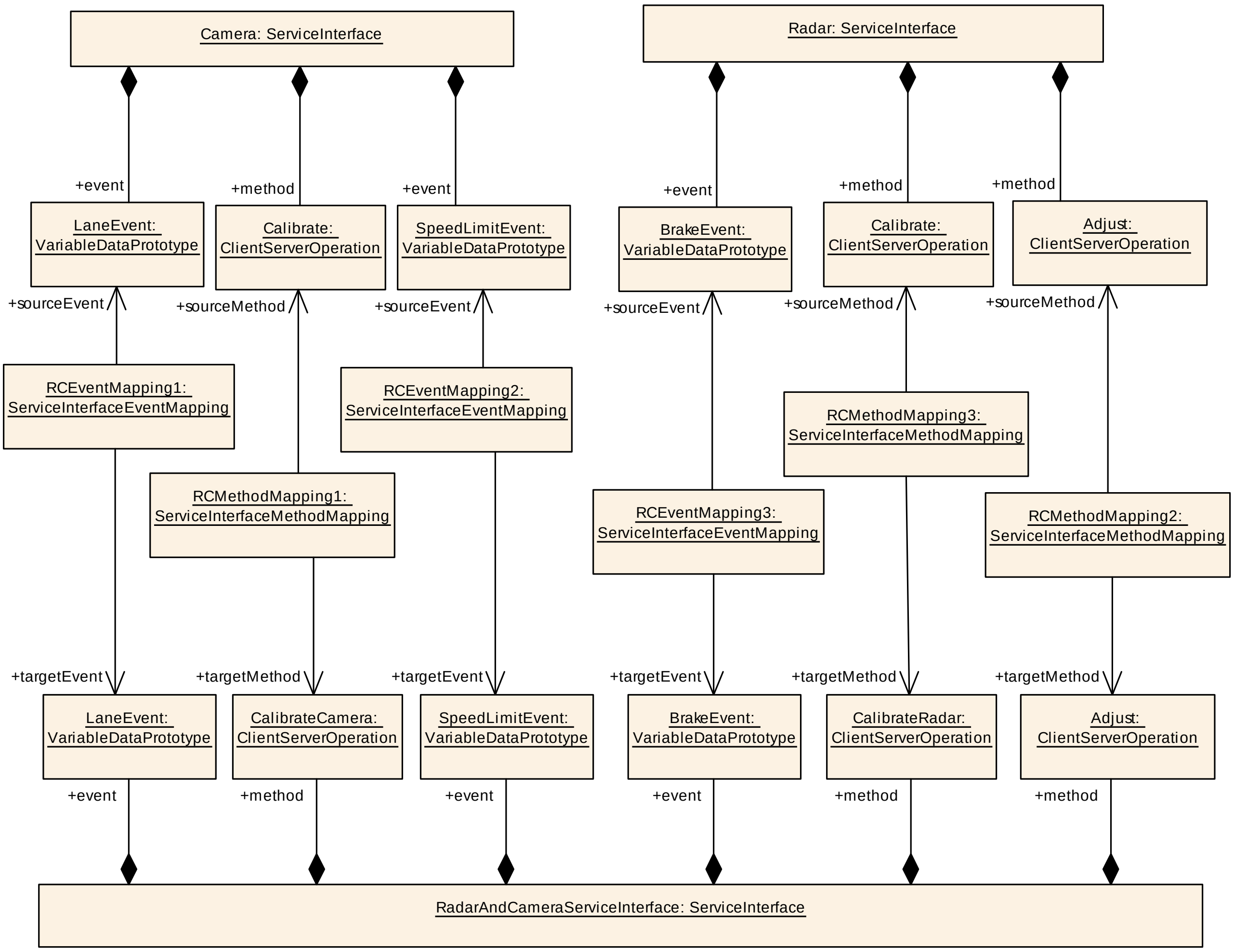


Figure A.10: Service Interface Element Mapping example

The combined `ServiceInterface` is offered over the network as a SOME/IP Service. Figure A.11 shows the assignment of the SOME/IP `serviceInterfaceId` to 31.

In addition SOME/IP `eventId`s are assigned to the `event`s and `methodId`s are assigned to the `method`s. Furthermore a single `SomeipEventGroup` is defined to which all `SomeipEventDeployment`s of the *RadarAndCamera* `ServiceInterface` are assigned.

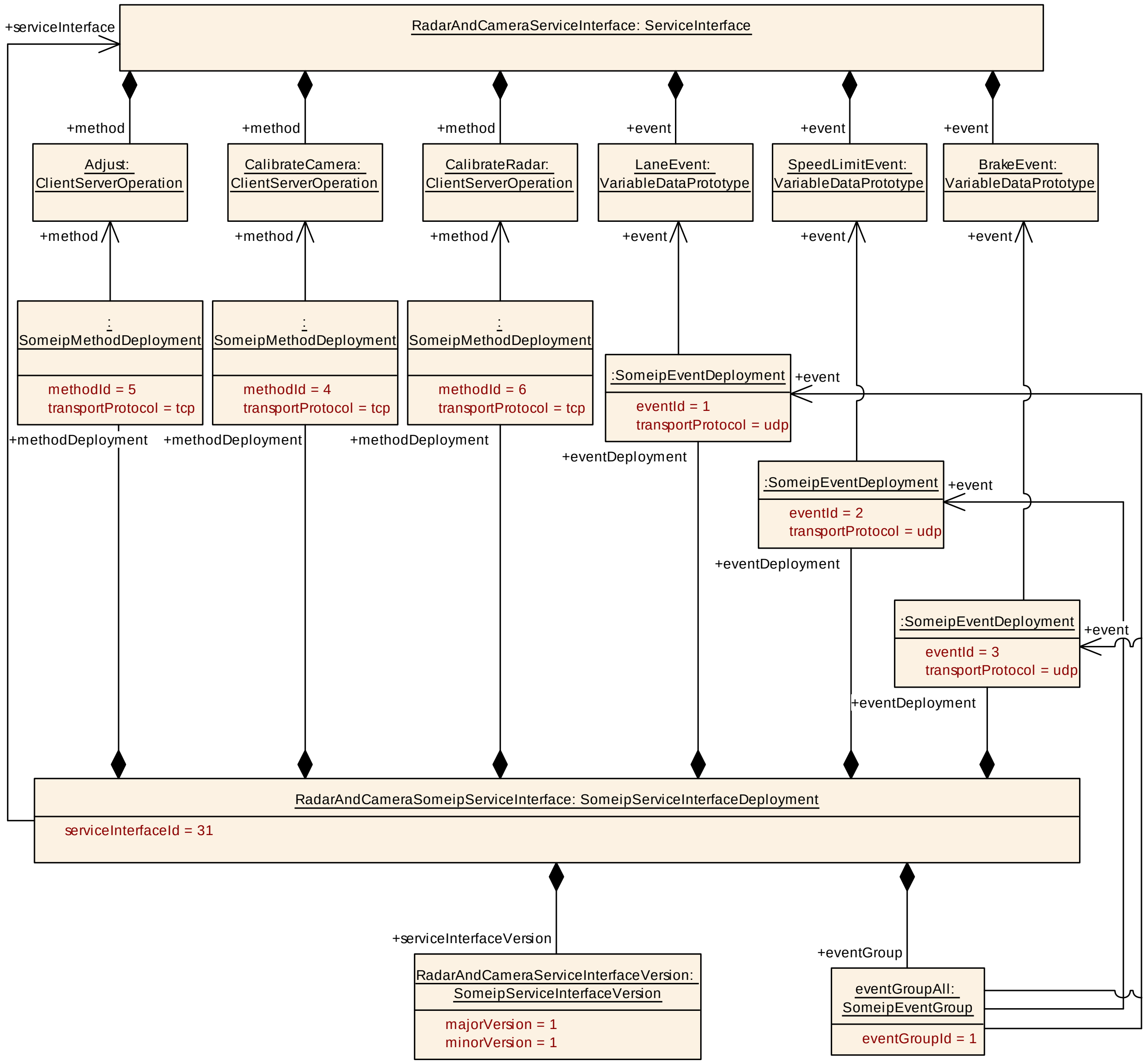


Figure A.11: SOME/IP Deployment

Figure A.12 shows a modeled `ProvidedSomeipServiceInstance` that is mapped to a `Machine`.

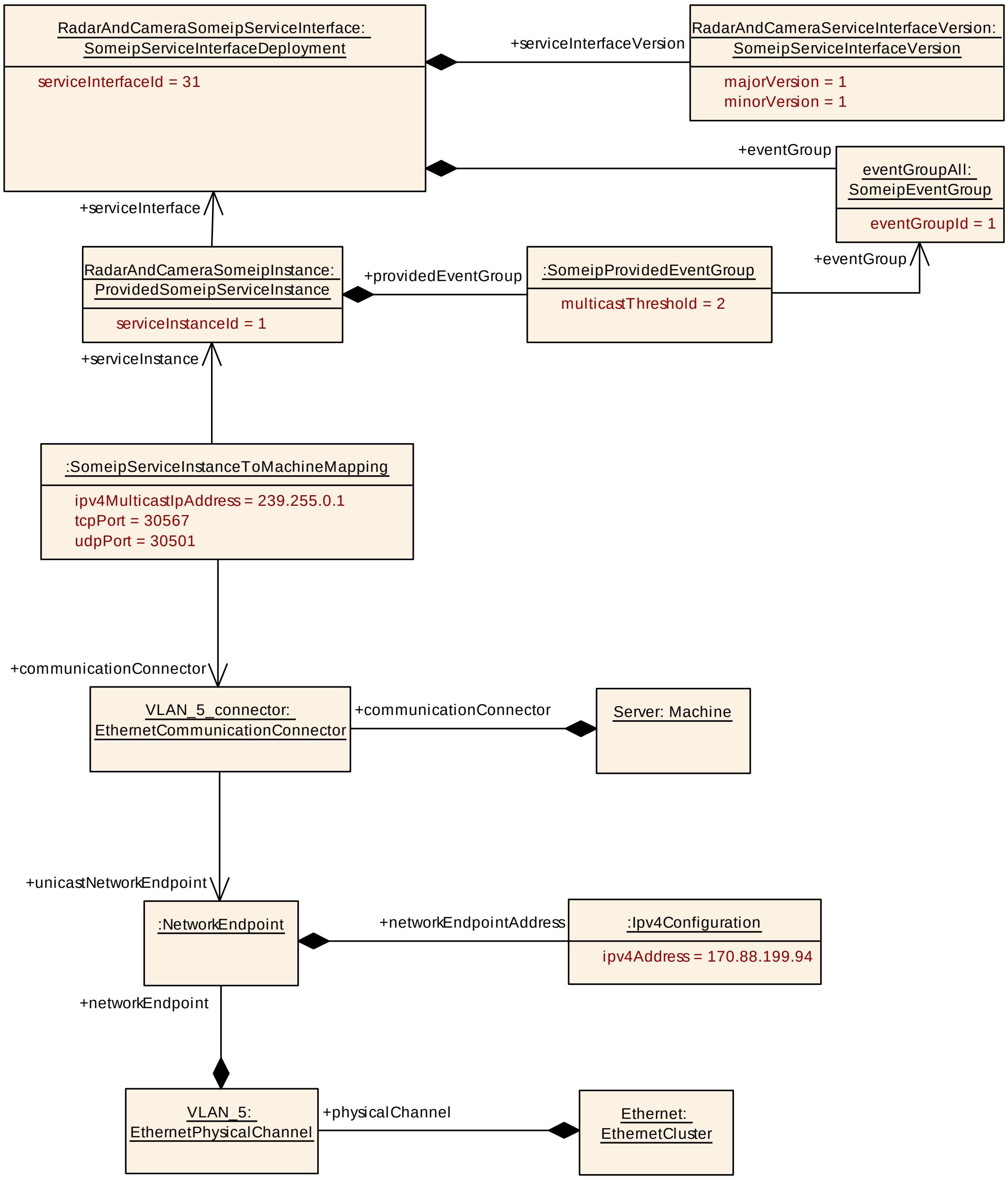


Figure A.12: SOME/IP Provided Service Instance

The displayed configuration in figure A.12 leads to a SOME/IP OfferService Message with the following content:

- ServiceId => `serviceInterfaceId` = 31
- InstanceId => `serviceInstanceId` = 1
- MajorVersion => 1
- MinorVersion => 1
- TTL => 3
- IPv4 Endpoint Option with IPv4 Address (170.88.199.94), Protocol (TCP), Port-Number (30567)

- IPv4 Endpoint Option with IPv4 Address (170.88.199.94), Protocol (UDP), Port-Number (30501)
- IP Multicast Endpoint Option with IPv4 Address (239.255.0.1), Protocol (UDP), PortNumber (30502)

An example of a RequiredSomeipServiceInstance is shown in Figure A.13.

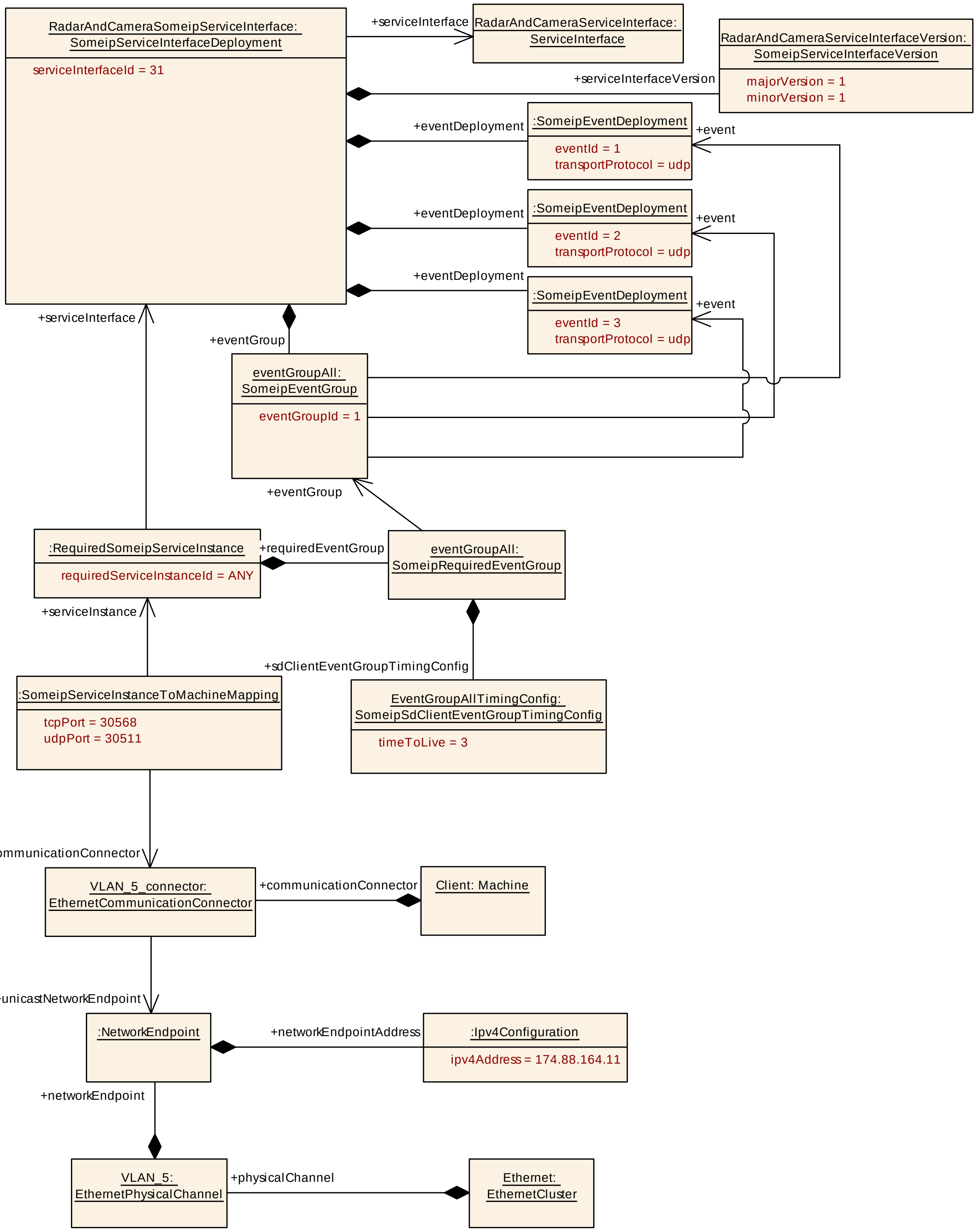


Figure A.13: SOME/IP Required Service Instance

The displayed configuration in figure A.13 leads to a SOME/IP Find Service Message with the following content:

- ServiceId => `serviceInterfaceId` = 31

- InstanceId => RequiredSomeipServiceInstance.requiredServiceInstanceId = ANY
- MajorVersion => majorVersion = 1
- MinorVersion => minorVersion = 1
- TTL => RequiredSomeipServiceInstance.sdClientConfig.serviceFindTimeToLive = 3

The displayed configuration in figure A.12 also leads to a SOME/IP SubscribeEventGroup Message content that is sent from the Service Requester to the Service Provider:

- ServiceId => taken from the OfferMessage
- InstanceId => taken from the OfferMessage
- MajorVersion => taken from the OfferMessage
- MinorVersion => taken from the OfferMessage
- Eventgroup ID => RequiredSomeipServiceInstance.requiredEventGroup.eventGroup.eventGroupId = 1
- TTL => RequiredSomeipServiceInstance.requiredEventGroup.sdClientEventGroupTimingConfig.timeToLive = 3
- IPv4 Endpoint Option with IPv4 Address (170.88.164.11), Protocol (UDP), Port-Number (30511)

A.6 Definition of Persistent Data

This chapter contains examples for the modeling of persistent data and file storage starting form the design aspect down to the definition of the persistent storage and the mapping between design and deployment.

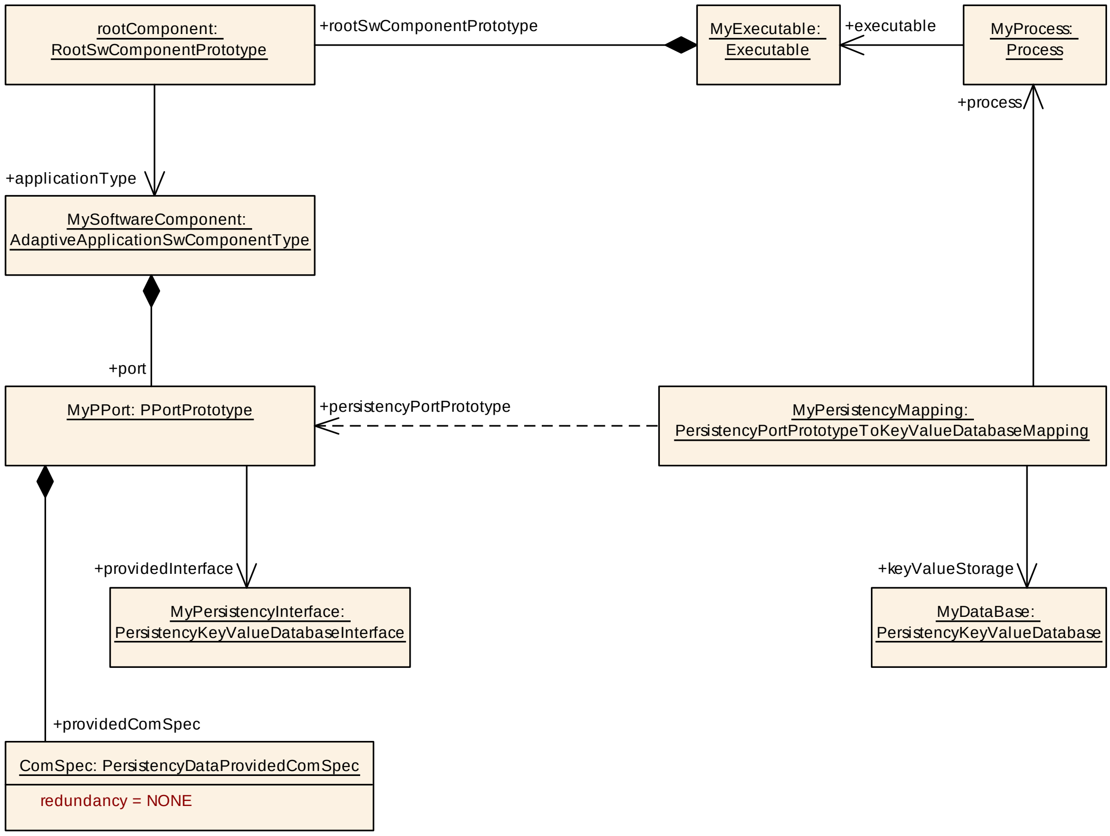


Figure A.14: Simple example modeling of persistent data (design + deployment)

The setup presented in Figure A.14 represents a case with reduced modeling of persistent data.

It is possible to extend the modeling to a deeper level of detail and also formally describe the individual data that is subject to persistency on both design and deployment level, see Figure A.15.

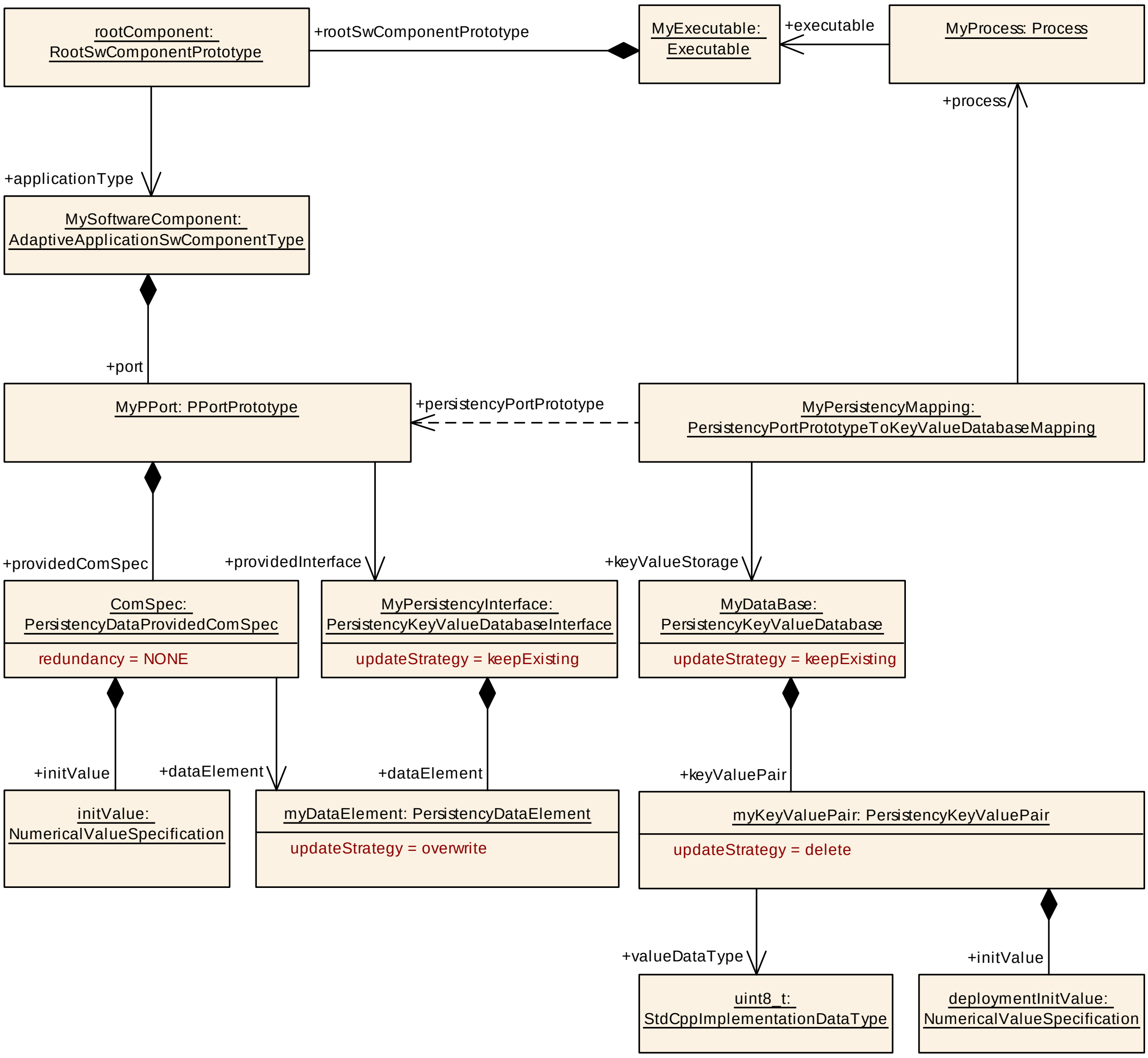


Figure A.15: Advanced example modeling of persistent data (design + deployment)

A.7 Definition of Persistent File

The setup presented in Figure A.16 represents a case with reduced modeling of persistent files.

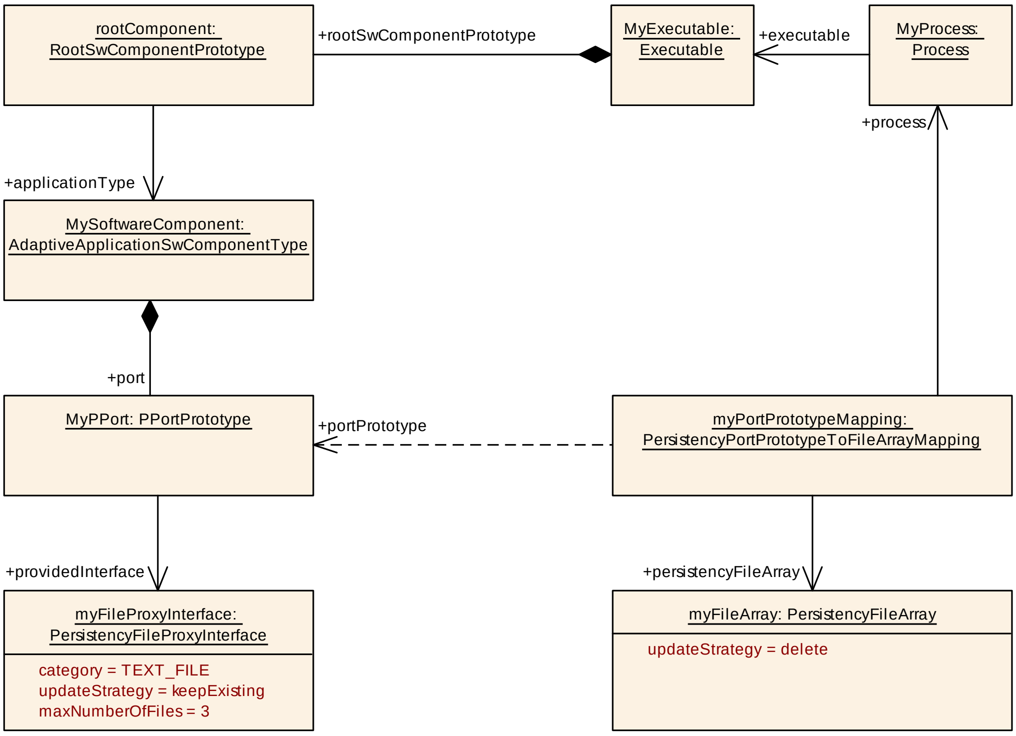


Figure A.16: Simple example modeling of persistent file (design + deployment)

It is possible to extend the modeling to a deeper level of detail and also formally describe the individual file that is subject to persistency on both design and deployment level, see Figure A.17.

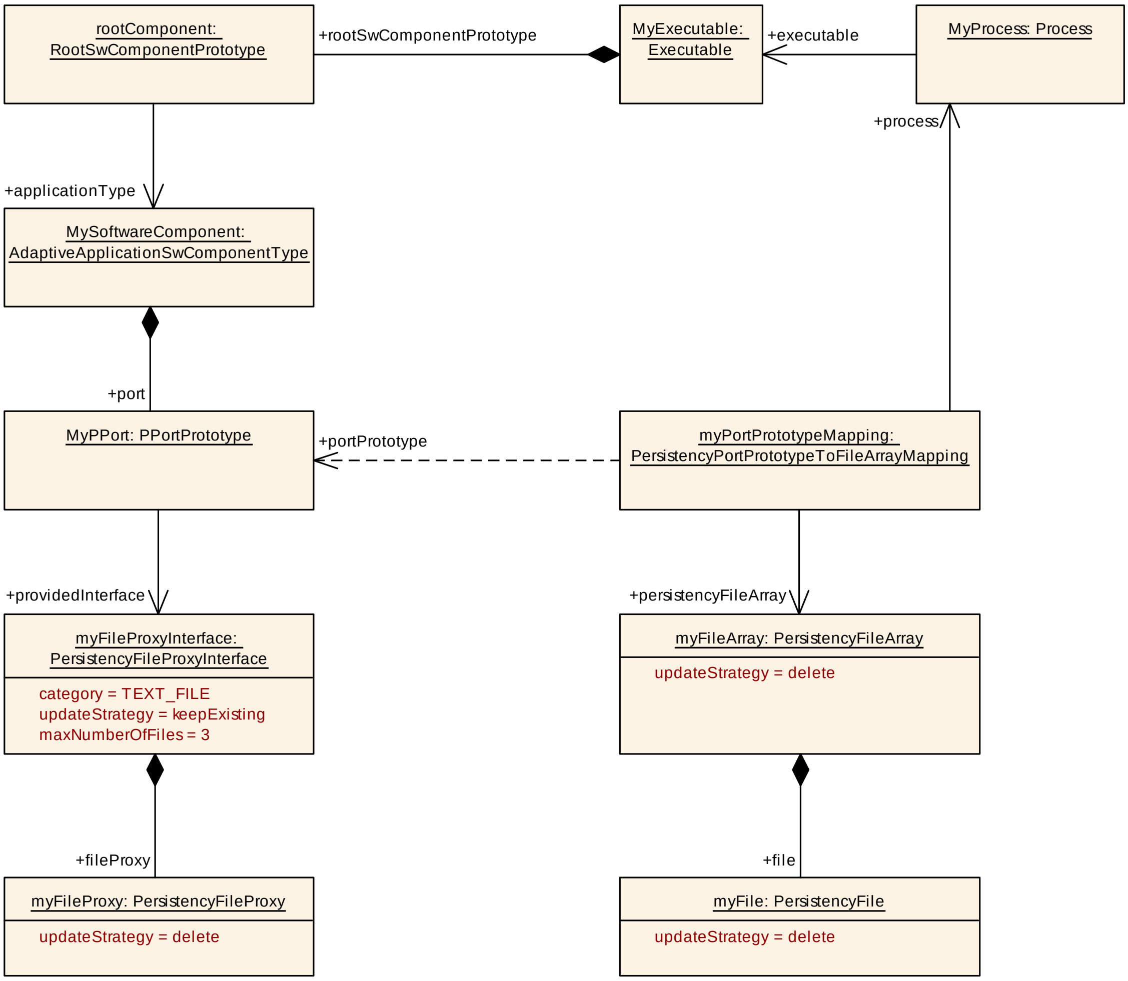


Figure A.17: Advanced example modeling of persistent file (design + deployment)

A.8 Definition of Phm interaction

This chapter contains examples for the modeling of platform health management. The example is structured into Application design and platform health management configuration.

A.8.1 Phm Application Design example

The simple example provided in figure A.18 shows the definition of a `PhmHealthChannelInterface` and a `PhmSupervisedEntityInterface`. This example will also be used in the subsequent section to define the platform health management configuration.

The `PhmHealthChannelInterface` *HealthChannel_A* defines two status attributes:

- *Good*

- *Bad*

The PhmSupervisedEntityInterface *SupervisedEntity_B* defines two checkpoints:

- *CP1*
- *CP2*

The AdaptiveApplicationSwComponentType *AdaptiveApplication* defines two RPortPrototypes

- *Hc_A* typed by *HealthChannel_A*
- *Se_B* typed by *SupervisedEntity_B*

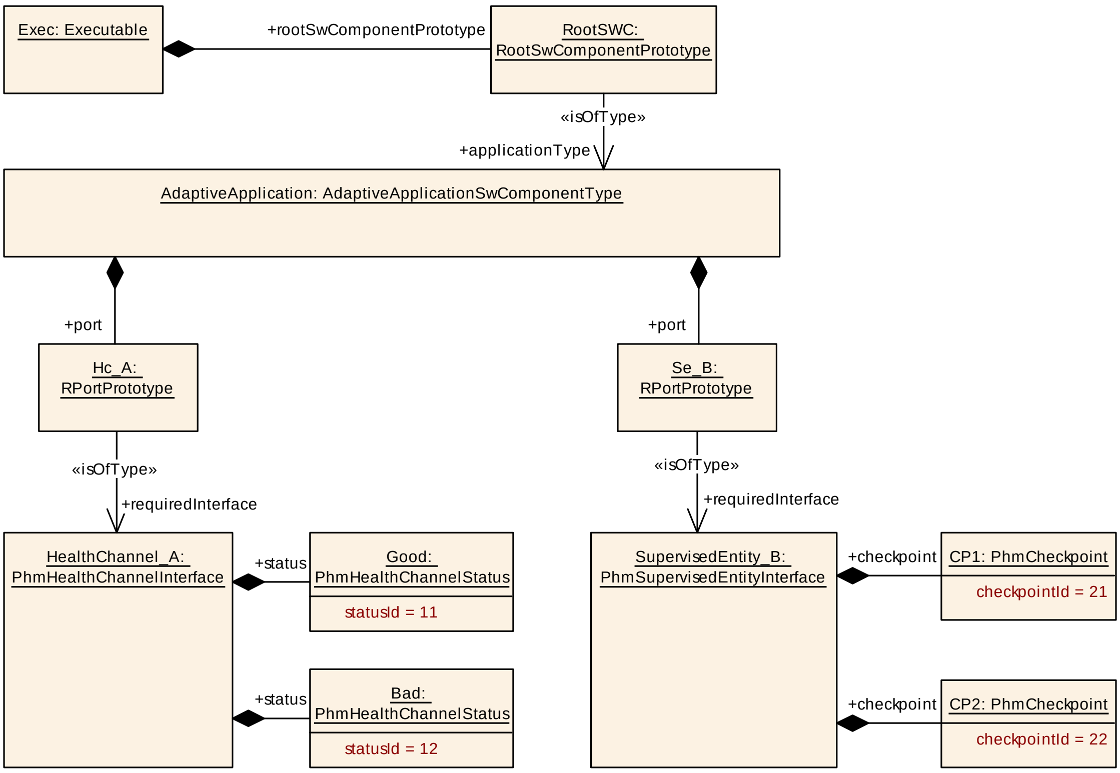


Figure A.18: Example modeling of Health Channel and Supervised Entity

A.8.2 Phm configuration example

When defining the configuration contribution for Phm it is required to first create representatives of the application design model artifacts (health channel status and supervised entity checkpoints) in the Phm configuration context. This is shown in figure A.19.

In this example the *PHM* PlatformHealthManagementContribution defines placeholder elements which refer to the respective application design model artifacts:

Example health channel:

- *Hc_Status_Good* refers to the *Good* status of *HealthChannel_A*
- *Hc_Status_Bad* refers to the *Bad* status of *HealthChannel_A*

Example supervision checkpoint:

- *Se_B_Cp1* refers to the *CP1* checkpoint of *SupervisedEntity_B*
- *Se_B_Cp2* refers to the *CP2* checkpoint of *SupervisedEntity_B*

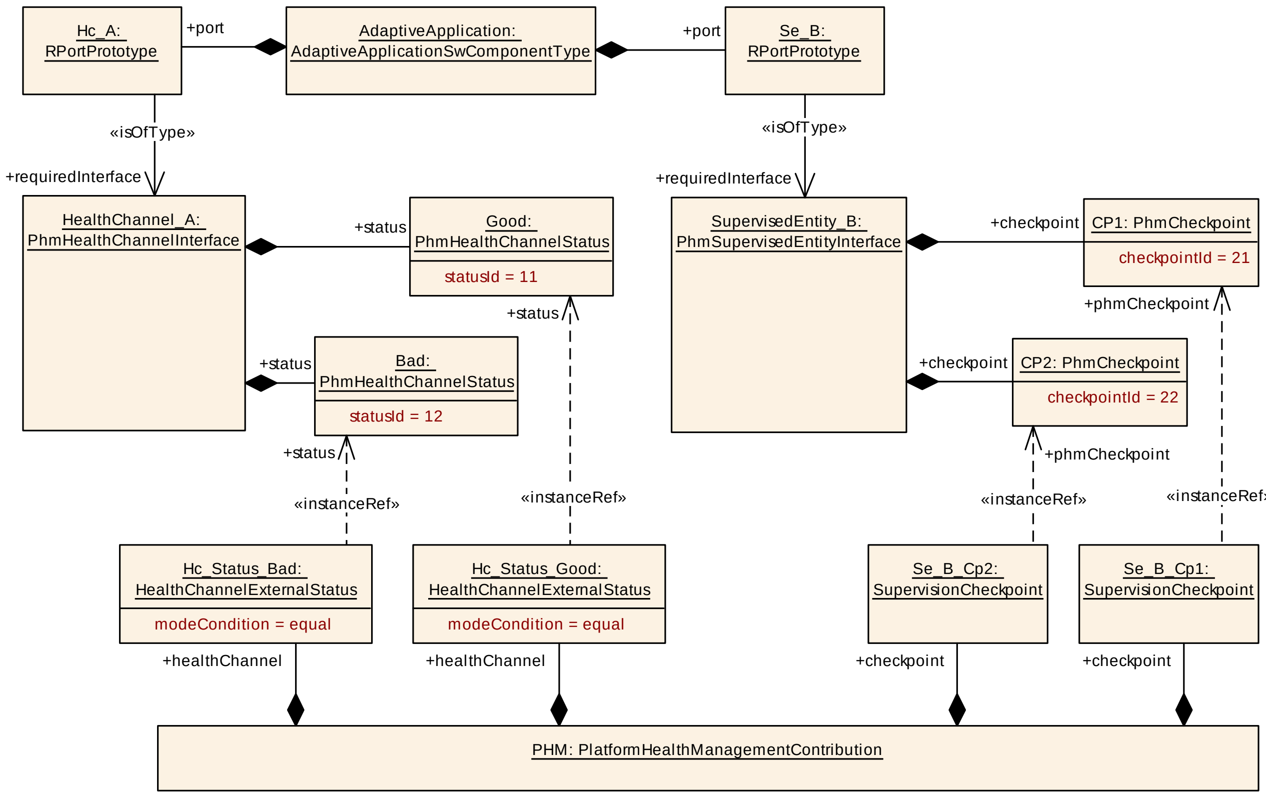


Figure A.19: Example modeling of Phm placeholder definition

Note that these instance references have a composite nature, which is shown in example figure A.20.

Here it is shown that in order to instance reference from the `HealthChannelExternalStatus` *Hc_Status_Bad* to the `PhmHealthChannelStatus` *Bad* there is the structured reference required consisting of

- `contextRootSwComponentPrototype`
- `contextRPortPrototype`
- `targetStatus`

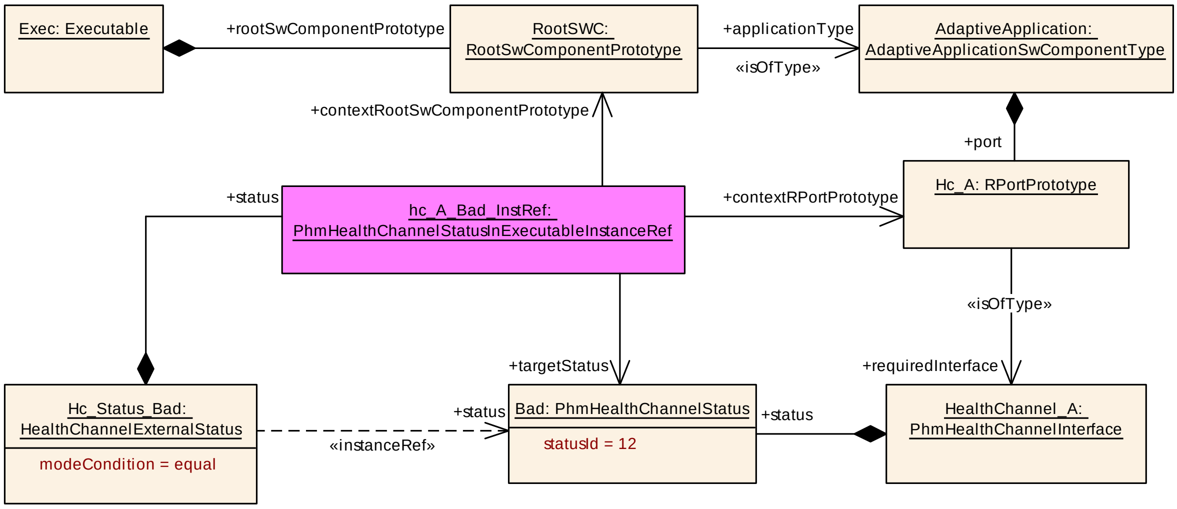


Figure A.20: Example modeling of Phm instance reference

The configuration of expressions is then based on the available placeholders of figure A.19.

One part of the example `PhmLogicalExpression` *Expr_1* is the already defined `HealthChannelExternalStatus` *Good* with the `modeCondition` == `equal`.

The second part of the example `PhmLogicalExpression` *Expr_1* is the evaluation of the supervision status of the `AliveSupervision` *AliveSup_1* which monitors the periodic reporting of the `SupervisionCheckpoint` *Se_B_Cp1*.

The result of *Se_B_Cp1* is taken as input to the `GlobalSupervision` *GlobalSup_1*.

And the `GlobalSupervision` *GlobalSup_1* is taken as input to the `HealthChannelSupervision` *HcSup_1* which in turn is the second input to the `PhmLogicalExpression` *Expr_1*.

The `PhmLogicalExpression` *Expr_1* then takes the two inputs and logically *AND*s them.

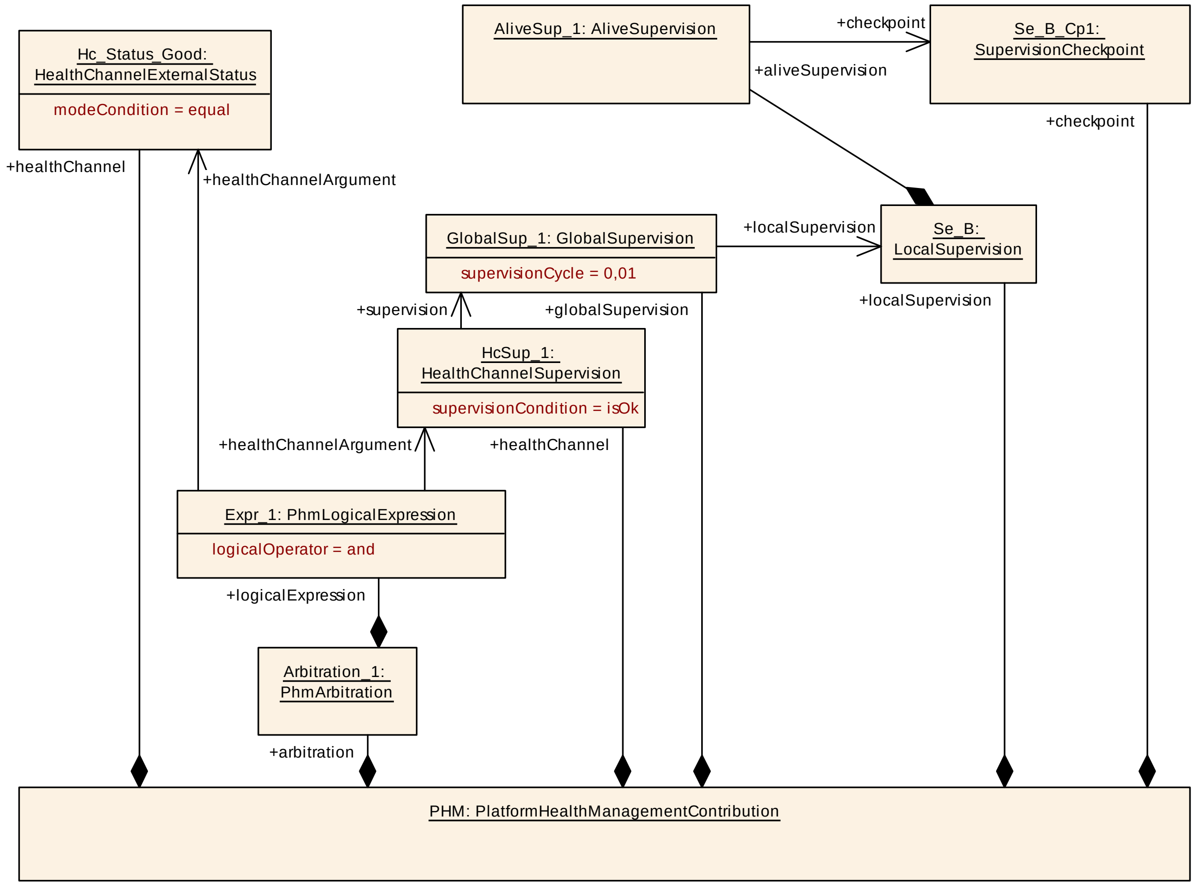


Figure A.21: Example modeling of Phm expression configuration

A.9 Scenarios to define a Vector

This section contains a non-comprehensive list of possible scenarios for the definition of a `CppImplementationDataType` of `category VECTOR`.

Please note that the general information contained in this chapter does not exclusively apply on to the vector data type. The latter has been picked as an arbitrary example for the visualization of the effect of configuration settings on the language binding.

Consequently, there is no further discussion of this topic with respect to a different kind of container data type.

Please note that for these example scenarios the namespace of a `CustomCppImplementationDataType` is assumed to be set to `x::y` and the `shortName` is assumed to be set to `CustVec`.

The `shortName` of a `StdCppImplementationDataType` is assumed to be set to `MyVec`.

If a custom `Allocator` is used in a scenario the value of `Allocator.shortName` shall be assumed to have the value `CustAlloc`.

Scenario	array size	custom allocator	custom type	Resulting C++ Code
I	No	No	No	`using MyVec = ara::core::Vector<std::uint8_t>`
II	Yes	No	No	`using MyVec = ara::core::Vector<std::uint8_t> //generator warning`
III	Yes	Yes	No	`using MyVec = ara::core::Vector<std::uint8_t, CustAlloc<std::uint8_t, MaxSize>>`
IV	No	Yes	No	`using MyVec = ara::core::Vector<std::uint8_t,CustAlloc<std::uint8_t>>`
V	Yes	Yes	Yes	`x::y::CustVec<ara::core::uint8_t, CustAlloc<std::uint8_t, MaxSize>>`
VI	Yes	No	Yes	`x::y::CustVec<std::uint8_t>`
VII	No	Yes	Yes	`x::y::CustVec<std::uint8_t, CustAlloc<std::uint8_t>>`
VIII	No	No	Yes	`x::y::CustVec<std::uint8_t>`

Table A.1: Example definitions of a CppImplementationDataType of category VECTOR

B General Modeling

This chapter has been created to explain model elements that are not directly related to specific design or deployment usage but have a more general scope. In other words, this chapter describes the structure and usage of some widely reusable modeling content.

B.1 Reference to a DataPrototype in a PortInterface

B.1.1 Reference to the inside of an ApplicationDataType

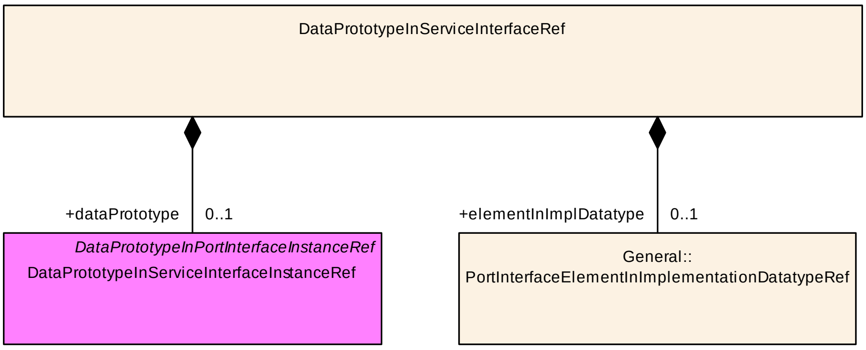


Figure B.1: Modeling of DataPrototypeInServiceInterfaceRef

[constr_1481]{DRAFT} **Usage of DataPrototypeInServiceInterfaceRef in the *AUTOSAR adaptive platform*** ⌈ If DataPrototypeInServiceInterfaceRef is used in the context of the *AUTOSAR adaptive platform* then the actual DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype shall be either a VariableDataPrototype or an ArgumentDataPrototype. ⌋*()*

Class	**DataPrototypeInServiceInterfaceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInPortInterfaceInstanceRef			
Note	This meta-class represents the ability to refer to an AUTOSAR DataPrototype in the context of a Service Interface. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataPrototype	DataPrototype	0..1	iref	This element represents the ability to: • refer to a DataPrototype in the context of a ServiceInterface. • refer to the internal structure of a DataPrototype in which is typed by an ApplicationDatatype the context of a ServiceInterface. **Tags:** atp.Status=draft

▽

△

Class	DataPrototypeInServiceInterfaceRef			
elementInImpl Datatype	PortInterfaceElementIn Implementation DatatypeRef	0..1	aggr	This element represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an ImplementationDatatype in the context of a Service Interface. **Tags:** atp.Status=draft

Table B.1: DataPrototypeInServiceInterfaceRef

Please note that the modeling of the reference to a `DataPrototype` in the context of a `PortInterface` can only be executed as the abstract template for concrete specializations because the abstract meta-class `PortInterface` does not aggregate a `DataPrototype` directly.

The abstract modeling of meta-class `DataPrototypeInPortInterfaceInstanceRef` is depicted in Figure B.2.

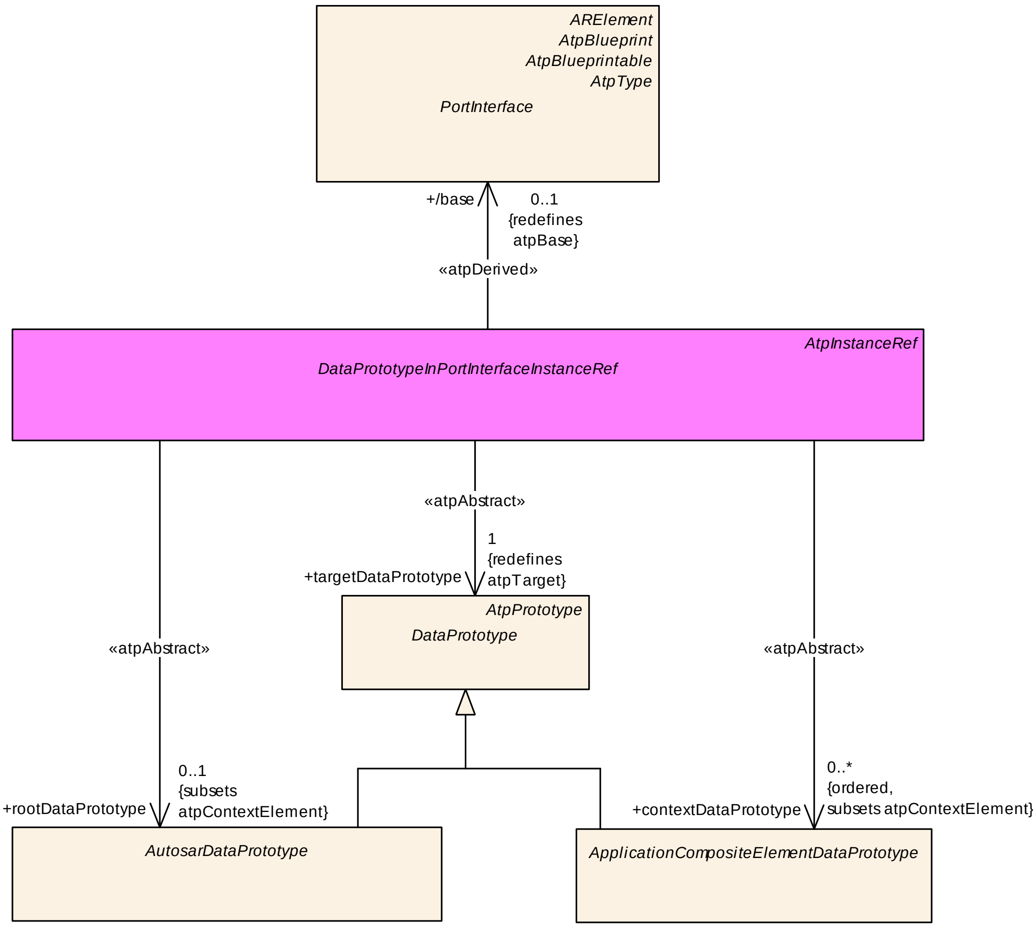


Figure B.2: Modeling of `DataPrototypeInPortInterfaceInstanceRef`

Class	***DataPrototypeInPortInterfaceInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInPortInterfaceInstanceRef			
Note	This meta-class represents the ability to: • refer to a DataPrototype in the context of a PortInterface. • refer to the internal structure of a DataPrototype which is typed by an ApplicationDatatype in the context of a PortInterface. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	DataPrototypeInServiceInterfaceInstanceRef			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
base	PortInterface	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft
contextDataPrototype (ordered)	ApplicationCompositeElementDataPrototype	*	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=20
rootDataPrototype	AutosarDataPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=10
targetDataPrototype	DataPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.2: DataPrototypeInPortInterfaceInstanceRef

The concrete specialization for the aggregation of a `DataPrototype` in the concrete `ServiceInterface` is depicted in Figure B.3.

The meta-class `DataPrototypeInServiceInterfaceInstanceRef` inherits from `DataPrototypeInPortInterfaceInstanceRef`.

The individual references modeled in the context of `DataPrototypeInServiceInterfaceInstanceRef` specialize the abstract structure defined in the context of `DataPrototypeInPortInterfaceInstanceRef`

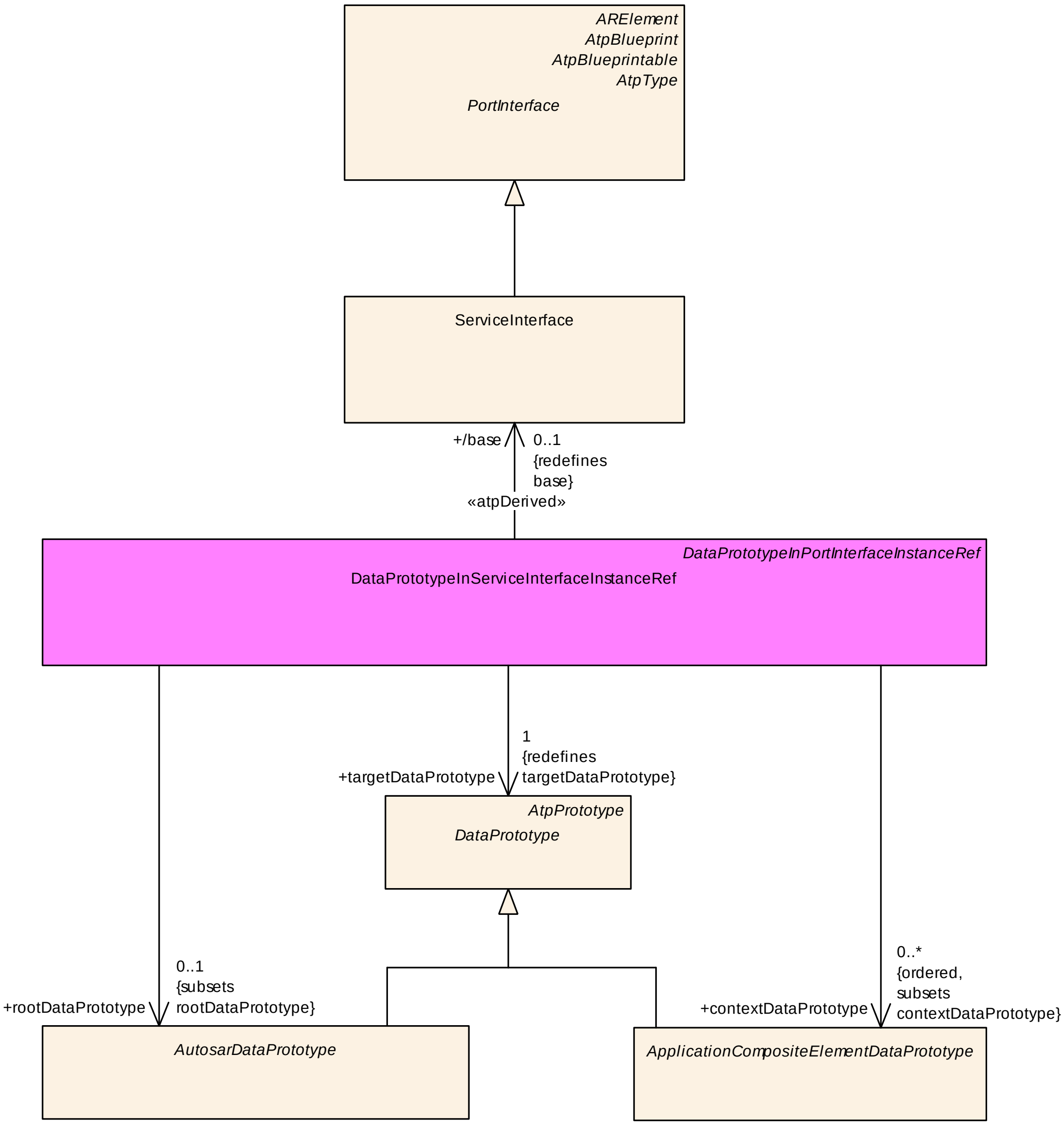


Figure B.3: Modeling of `DataPrototypeInServiceInterfaceInstanceRef`

Class	**DataPrototypeInServiceInterfaceInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInPortInterfaceInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*, *DataPrototypeInPortInterfaceInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	ServiceInterface	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft
contextDataPrototype (ordered)	ApplicationCompositeElementDataPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
rootDataPrototype	AutosarDataPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10

▽

△

Class	DataPrototypeInServiceInterfaceInstanceRef			
targetData Prototype	DataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.3: DataPrototypeInServiceInterfaceInstanceRef

B.1.2 Reference to the inside of a CppImplementationDataType

Please note that the modeling of instanceRef-like references into the internals of a CppImplementationDataType differs from the way how internals of an ImplementationDataType could be referenced.

In particular, references to context elements can be directed to ImplementationDataTypeElement because both arrays and structures are modeled by means of ImplementationDataTypeElement.

This approach has changed with the advent of CppImplementationDataType and therefore the same approach is not possible for CppImplementationDataTypeElement.

In the case of CppImplementationDataType, both CppImplementationDataTypeElement as well as CppImplementationDataType can become the target of a context reference. And since the context reference is supposed to be ordered it is simply not possible to straight up model two context references, one for CppImplementationDataType and one for CppImplementationDataTypeElement.

Instead, it is necessary to introduce an abstract base class named CppImplementationDataTypeContextTarget for both CppImplementationDataType and CppImplementationDataTypeElement and then **direct context references at the abstract base class**.

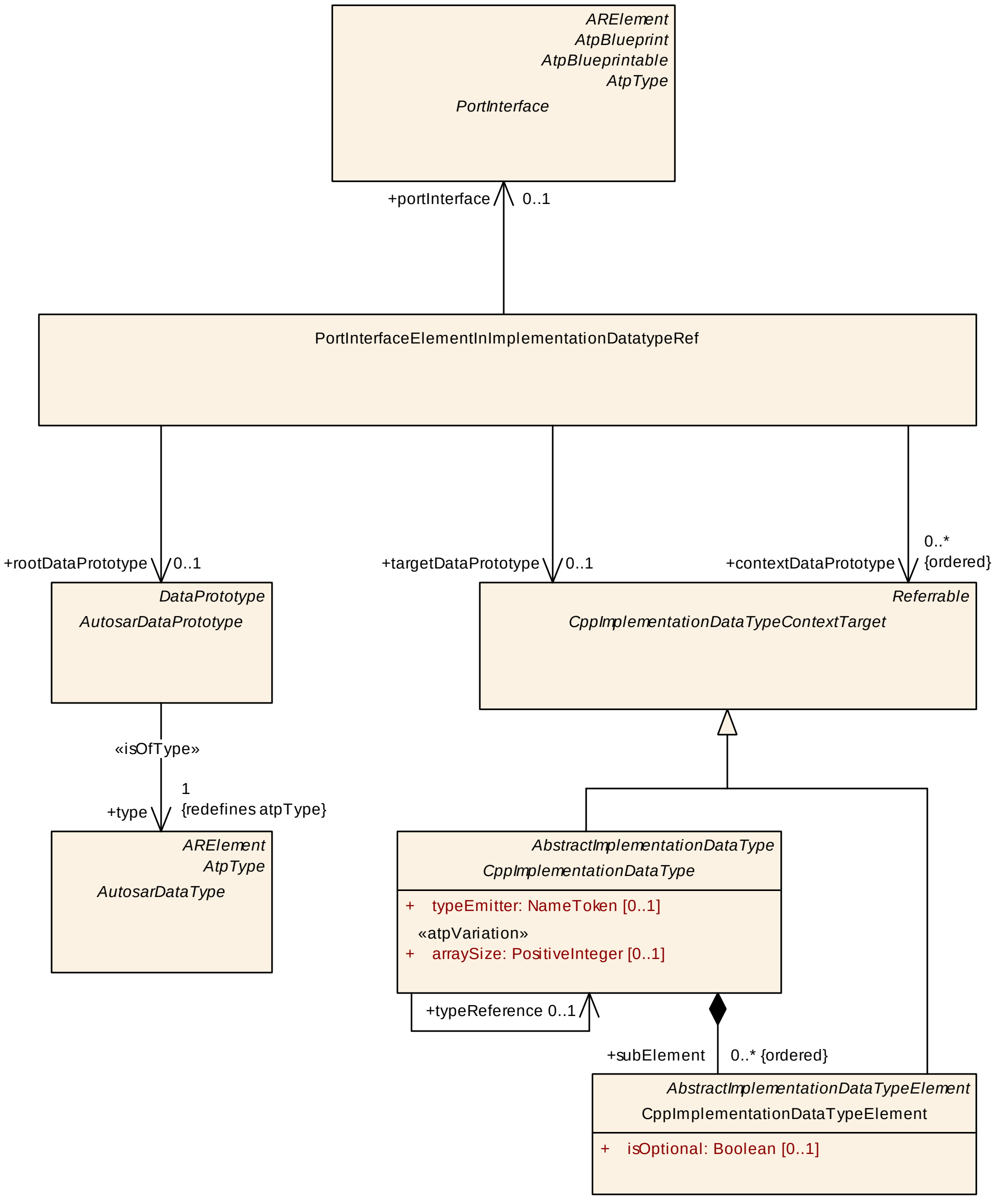


Figure B.4: Modeling of `PortInterfaceElementInImplementationDatatypeRef`

Class	**PortInterfaceElementInImplementationDatatypeRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an implementationDatatype in the context of a PortInterface. In other words, this meta-class shall not be used to model a reference to the **AutosarDataPrototype as a target itself**, **even** if the AutosarDataPrototype is typed by an ImplementationDataType **and even** if that ImplementationDataType represents a composite data type. **Tags:** atp.Status=draft			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*

▽

△

Class	**PortInterfaceElementInImplementationDatatypeRef**			
contextDataPrototype (ordered)	CppImplementationDataTypeContextTarget	*	ref	This is a context in case there are subelements with explicit types. The reference has to be ordered to properly reflect the nested structure. **Tags:** atp.Status=draft
portInterface	PortInterface	0..1	ref	This is the PortInterface that contains the the rootDataPrototype. **Tags:** atp.Status=draft
rootDataPrototype	AutosarDataPrototype	0..1	ref	This rootDataPrototype defines the AutosarDataPrototype in which the target can be found. **Tags:** atp.Status=draft
targetDataPrototype	CppImplementationDataTypeContextTarget	0..1	ref	This is the target reference to a subElement that is defined inside of the rootDataPrototype. **Tags:** atp.Status=draft

Table B.4: PortInterfaceElementInImplementationDatatypeRef

Class	***CppImplementationDataTypeContextTarget*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class has the ability to serve as the context in instanceRef-like modeling for CppImplementationDataType and CppImplementationDataTypeElement **Tags:** atp.Status=draft			
Base	*ARObject*, *Referrable*			
Subclasses	*CppImplementationDataType*, CppImplementationDataTypeElement			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table B.5: CppImplementationDataTypeContextTarget

B.2 Reference to a AutosarDataPrototype in an Executable

The creation of the meta-model for creating a reference to an `AutosarDataPrototype` in the context of an `Executable` is executed in a two-step approach where first an abstract structure of the reference is created.

The abstract structure is the basis for the refinement with respect to specific roles of `AutosarDataPrototype`s.

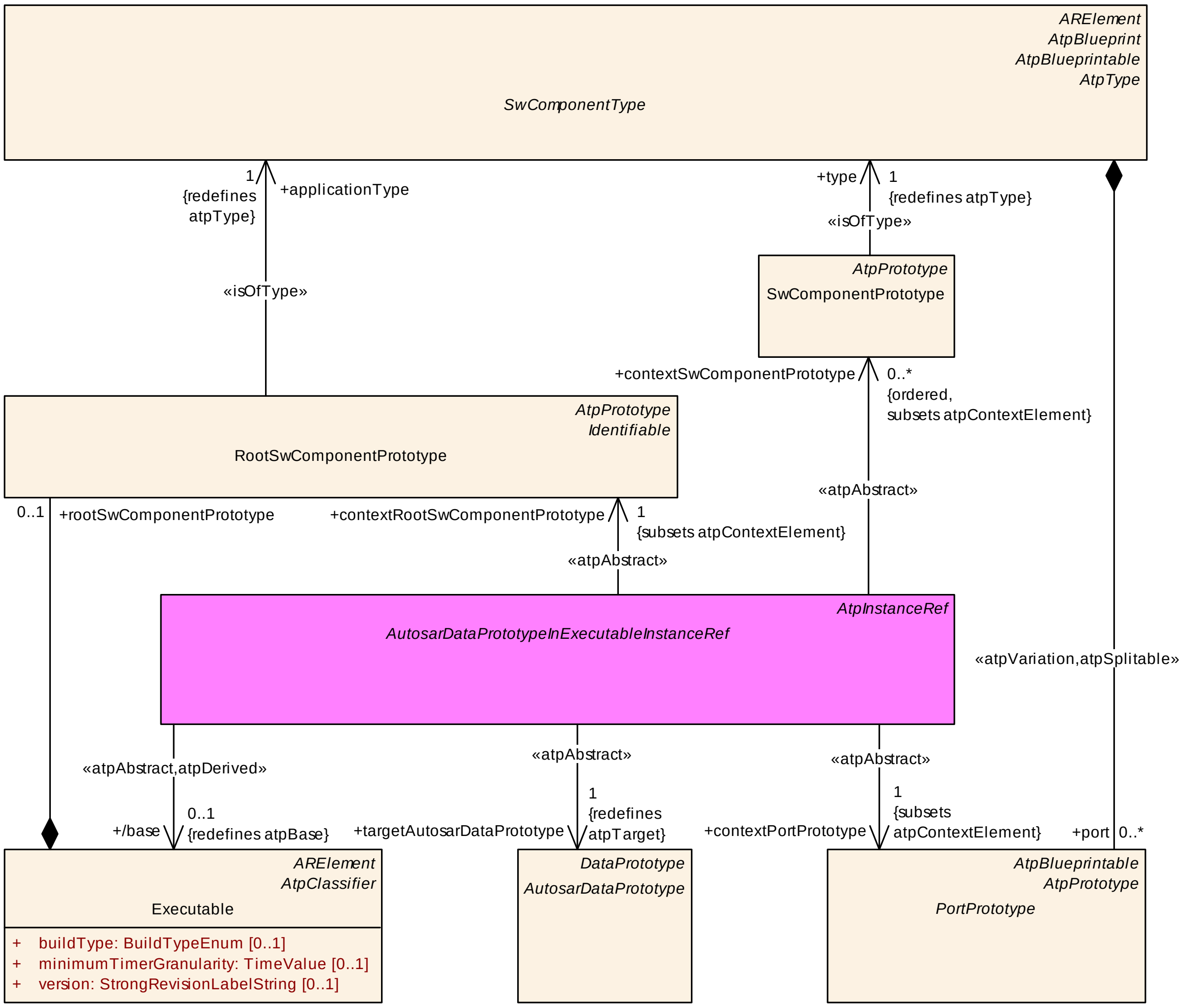


Figure B.5: Modeling of abstract AutosarDataPrototypeInExecutableInstanceRef

Class	***AutosarDataPrototypeInExecutableInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	EventInExecutableInstanceRef, FieldInExecutableInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived **Tags:** atp.Status=draft
contextPort Prototype	PortPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=10
contextSw ComponentPro-totype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=20

▽

△

Class	***AutosarDataPrototypeInExecutableInstanceRef*** (abstract)			
targetAutosarDataPrototype	AutosarDataPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.6: AutosarDataPrototypeInExecutableInstanceRef

Two specializations of `AutosarDataPrototypeInExecutableInstanceRef` exist:

- `EventInExecutableInstanceRef`
- `FieldInExecutableInstanceRef`

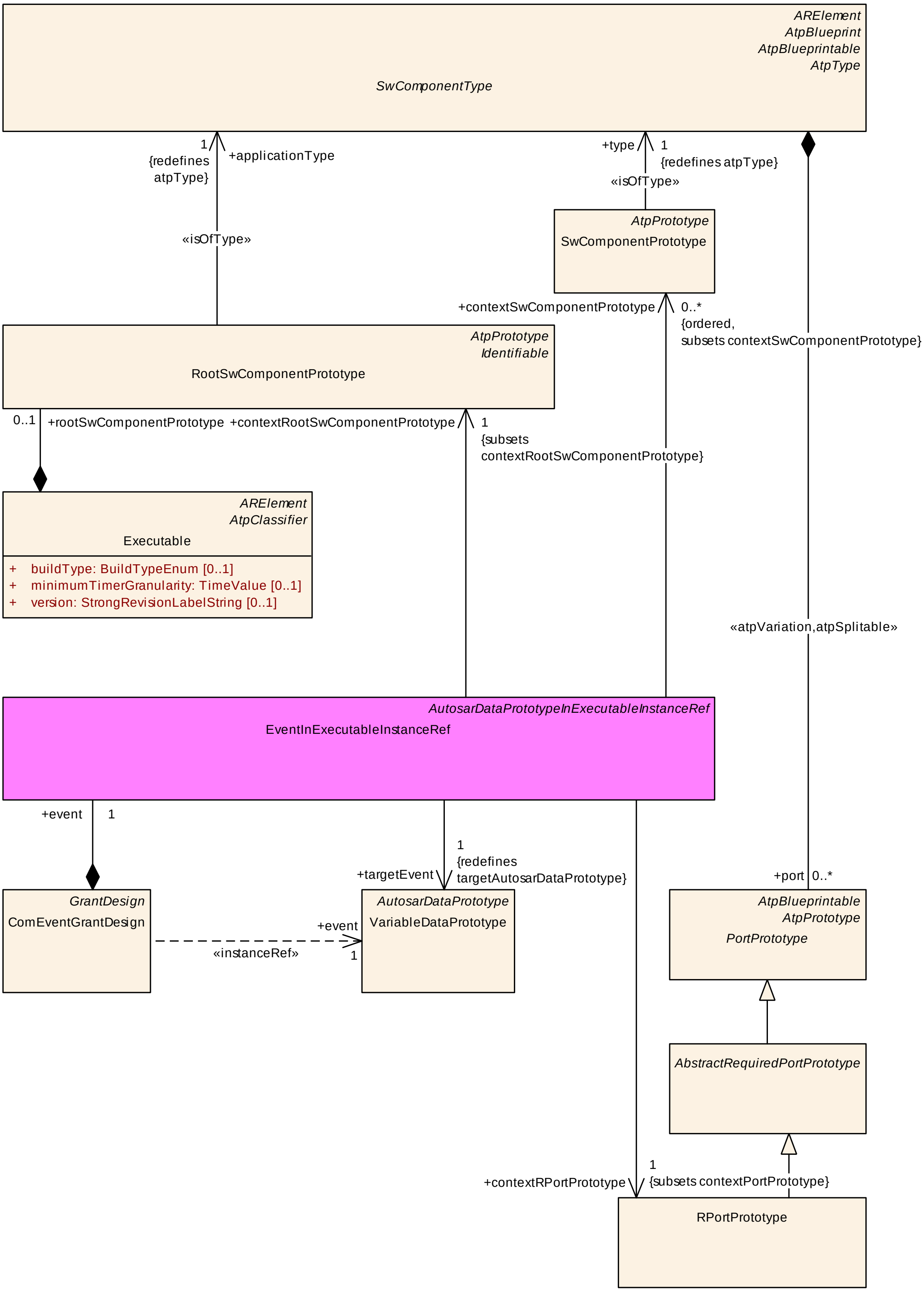


Figure B.6: Modeling of concrete `EventInExecutableInstanceRef` derived from `AutosarDataPrototypeInExecutableInstanceRef`

Class	EventInExecutableInstanceRef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*, *AutosarDataPrototypeInExecutableInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
contextRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
contextSw ComponentPro-totype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetEvent	VariableDataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.7: EventInExecutableInstanceRef

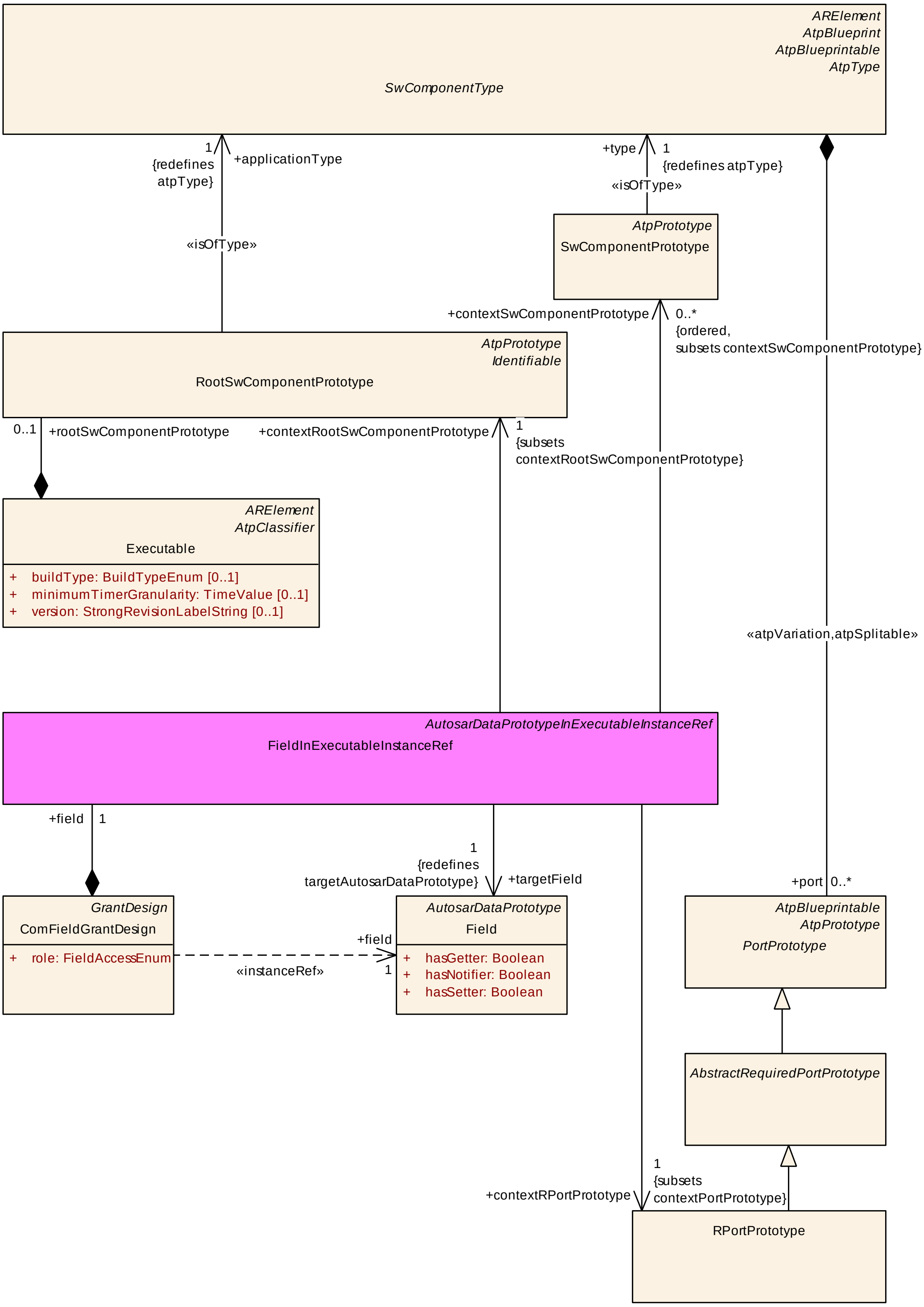


Figure B.7: Modeling of concrete `FieldInExecutableInstanceRef` derived from `AutosarDataPrototypeInExecutableInstanceRef`

Class	FieldInExecutableInstanceRef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*, *AutosarDataPrototypeInExecutableInstanceRef*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
contextRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
contextSw ComponentPro-totype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetField	Field	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.8: FieldInExecutableInstanceRef

B.3 Reference to a PortPrototype in an Executable

The creation of the meta-model for creating a reference to a `PortPrototype` in the context of an `Executable` is executed in a two-step approach where first an abstract structure of the reference is created.

The abstract structure is the basis for the refinement with respect to specific roles of `PortPrototype`s.

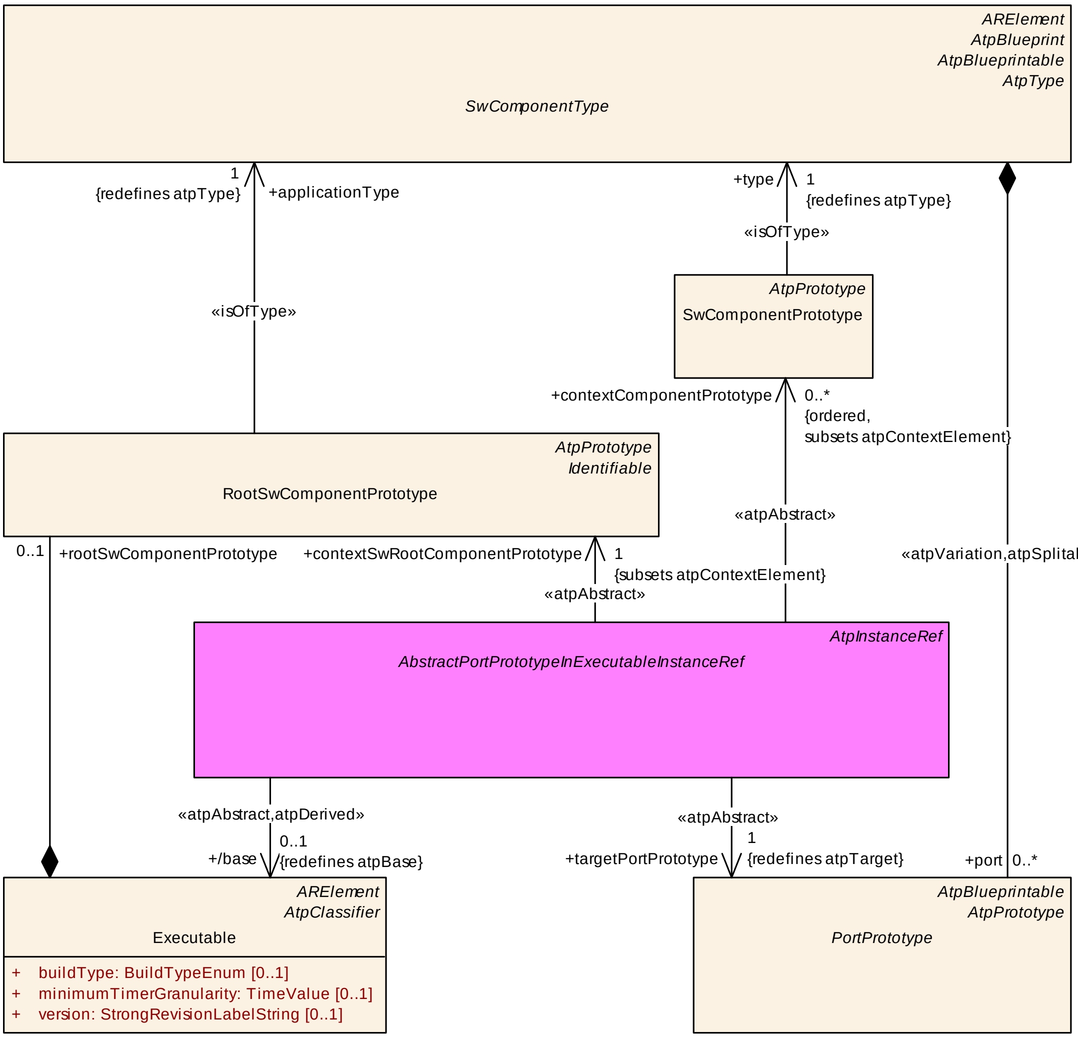


Figure B.8: Modeling of abstract AbstractPortPrototypeInExecutableInstanceRef

Class	***AbstractPortPrototypeInExecutableInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	PPortPrototypeInExecutableInstanceRef, PortPrototypeInExecutableInstanceRef, RPortPrototypeInExecutableInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived **Tags:** atp.Status=draft
contextComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=20
contextSwRootComponentPrototype	RootSwComponentPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=10

▽

△

Class	***AbstractPortPrototypeInExecutableInstanceRef*** (abstract)			
targetPort Prototype	PortPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.9: AbstractPortPrototypeInExecutableInstanceRef

Three specializations of `AbstractPortPrototypeInExecutableInstanceRef` exist:

- `PPortPrototypeInExecutableInstanceRef`
- `RPortPrototypeInExecutableInstanceRef`
- `PortPrototypeInExecutableInstanceRef`

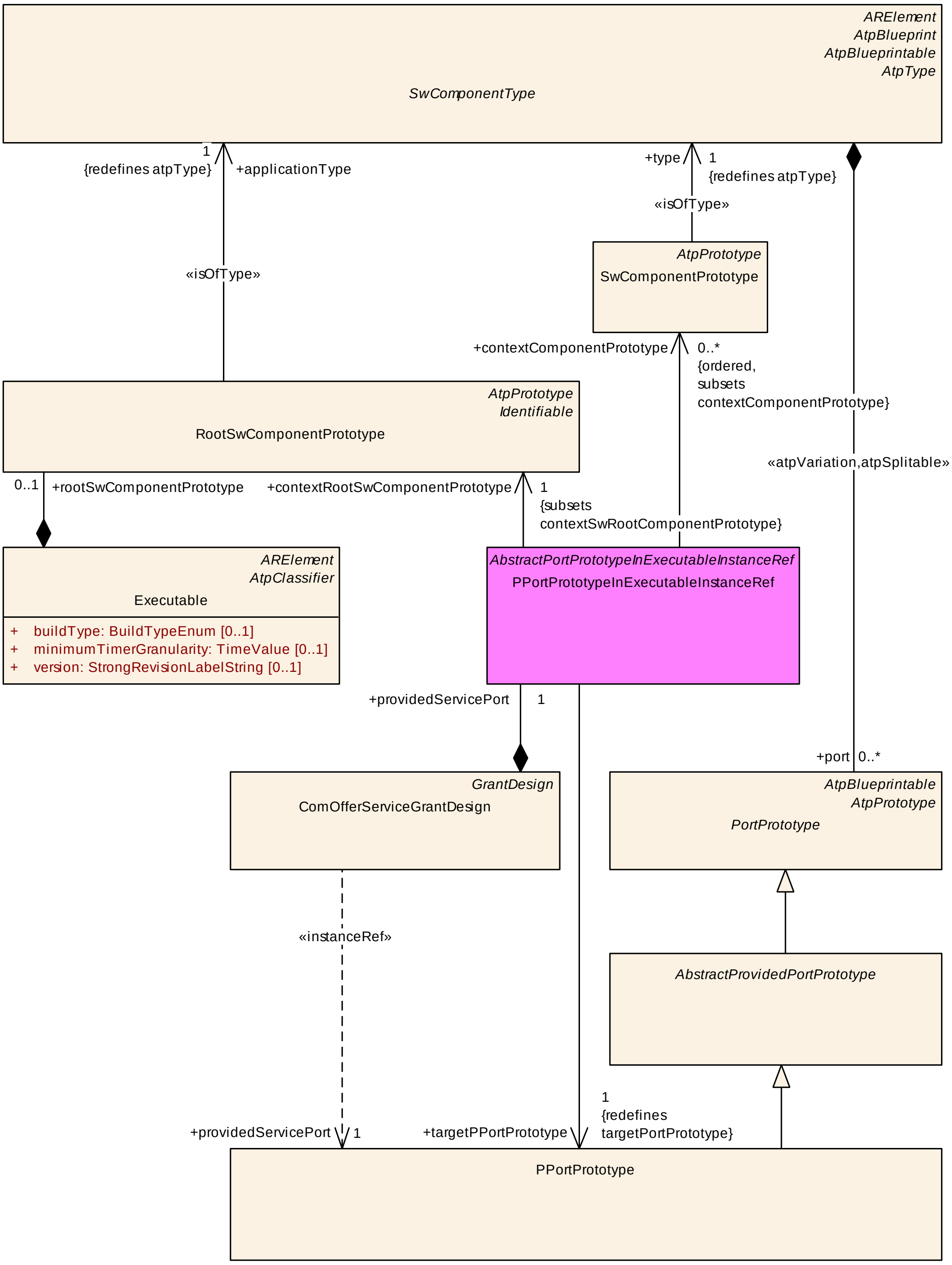


Figure B.9: Modeling of concrete `PPortPrototypeInExecutableInstanceRef` derived from `AbstractPortPrototypeInExecutableInstanceRef`

Class	PPortPrototypeInExecutableInstanceRef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
contextComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
contextRootSwComponentPrototype	RootSwComponentPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetPPortPrototype	PPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.10: PPortPrototypeInExecutableInstanceRef

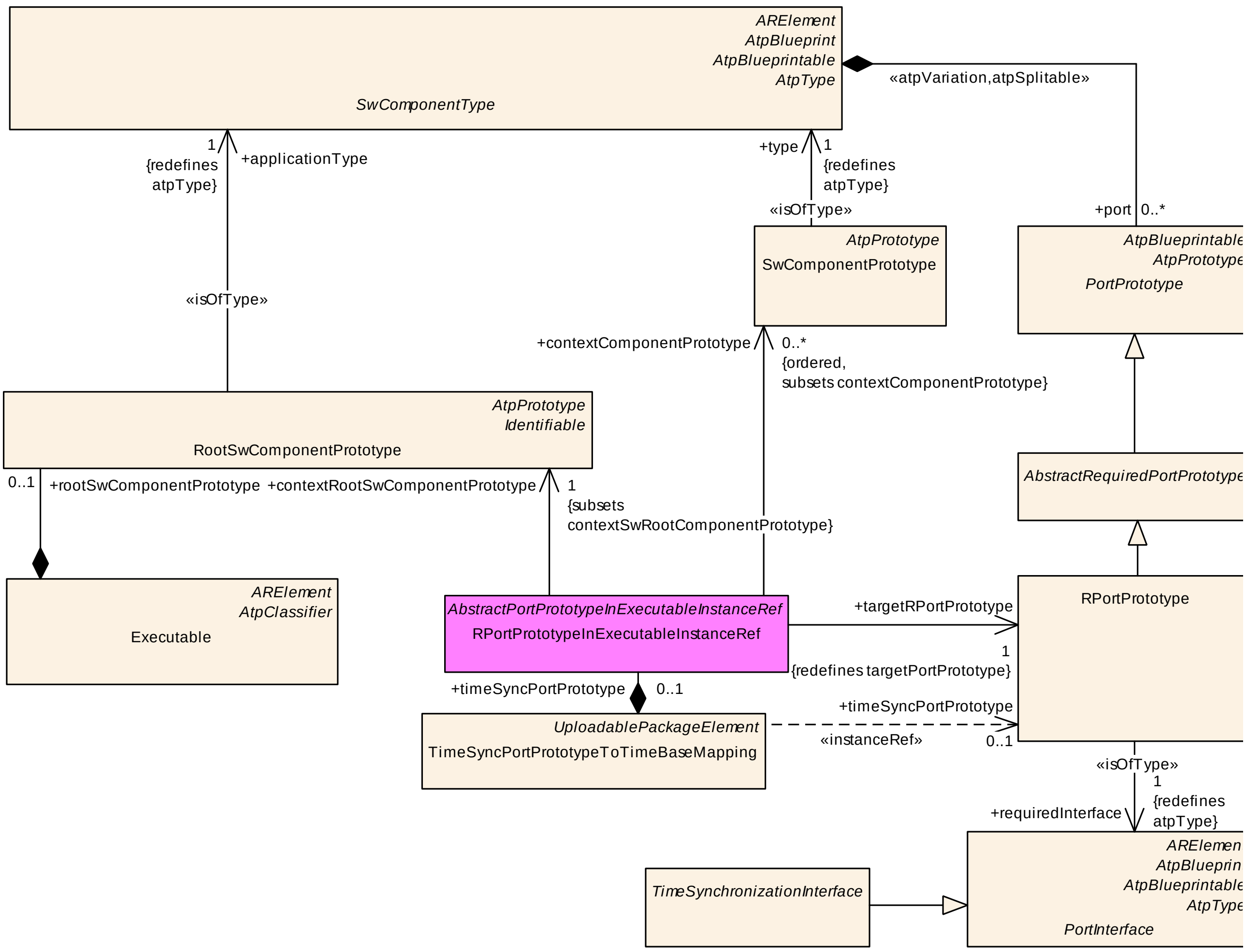


Figure B.10: Modeling of concrete `RPortPrototypeInExecutableInstanceRef` derived from `AbstractPortPrototypeInExecutableInstanceRef`

Class	**RPortPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
context ComponentPrototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.11: RPortPrototypeInExecutableInstanceRef

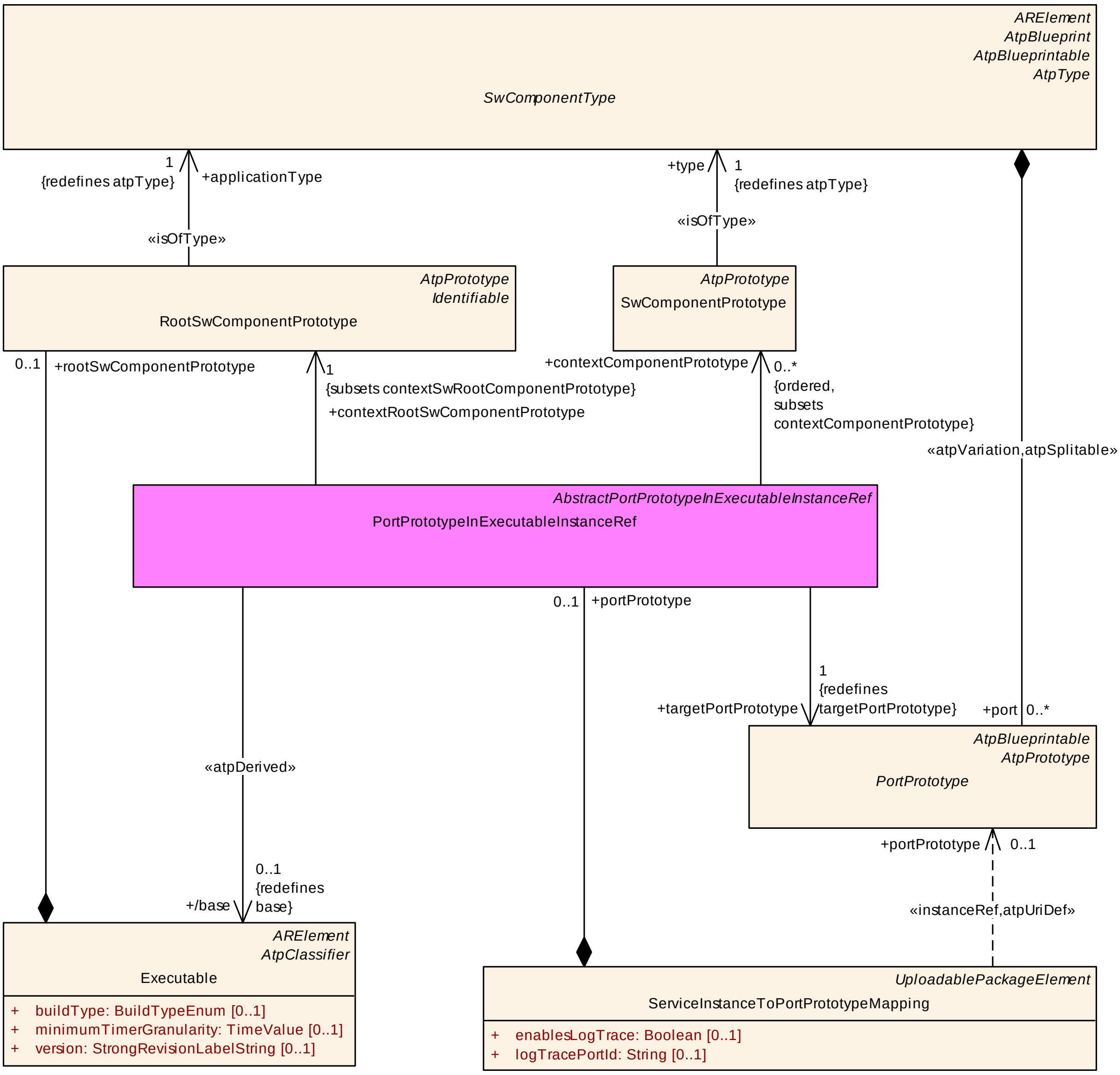


Figure B.11: Modeling of PortPrototypeInExecutableInstanceRef

Class	PortPrototypeInExecutableInstanceRef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest::InstanceRefs			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
contextComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSwComponentPrototype	RootSwComponentPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetPortPrototype	PortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.12: PortPrototypeInExecutableInstanceRef

B.4 Modeling of a Method in an Executable

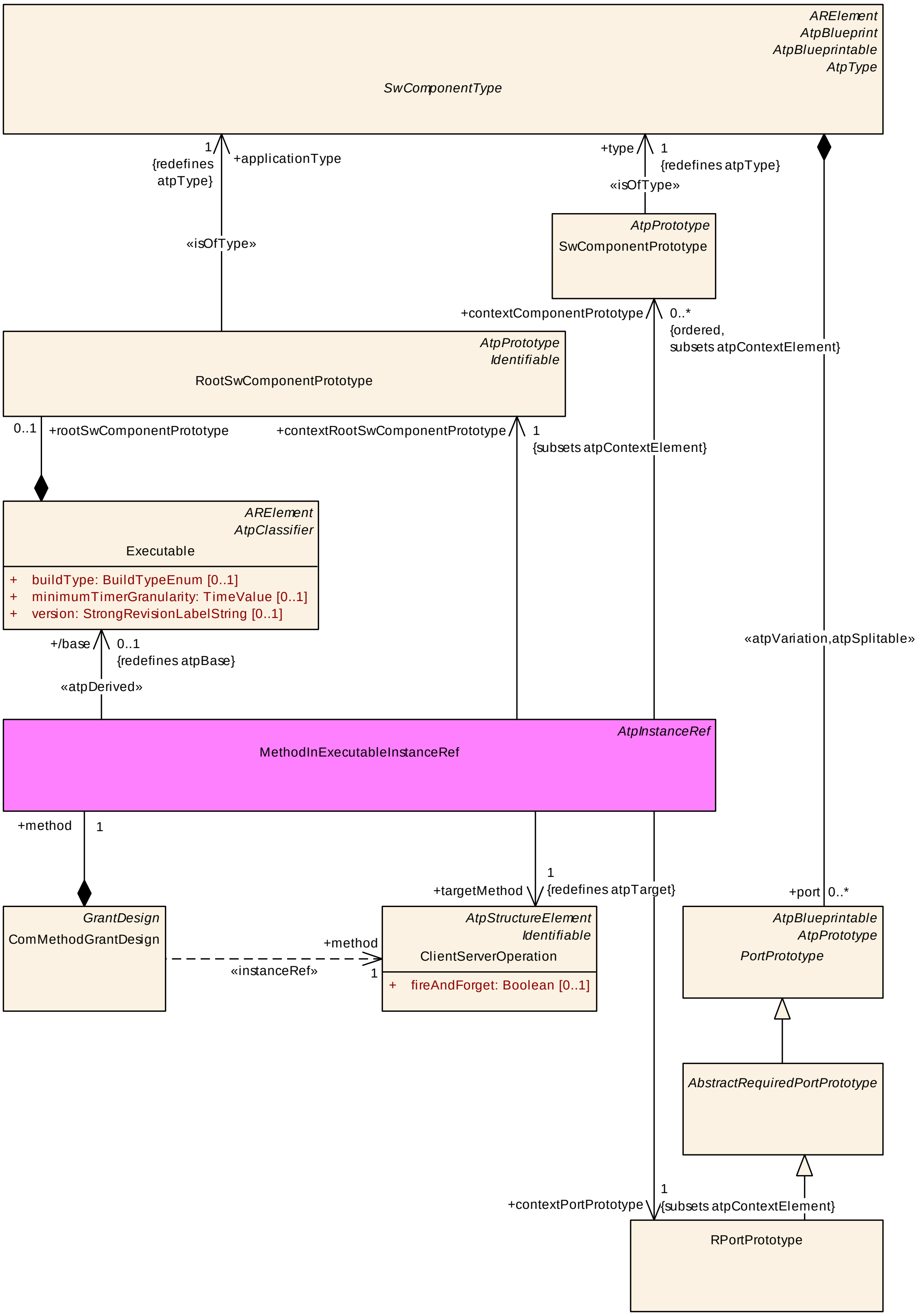


Figure B.12: Modeling of `MethodInExecutableInstanceRef`

Class	MethodInExecutableInstanceRef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft
context ComponentPrototype (ordered)	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
contextPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetMethod	ClientServerOperation	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.13: MethodInExecutableInstanceRef

B.5 Modeling of Mode-related InstanceRefs

This section illustrates the concrete modeling of the instance references used in the previous parts of this document.

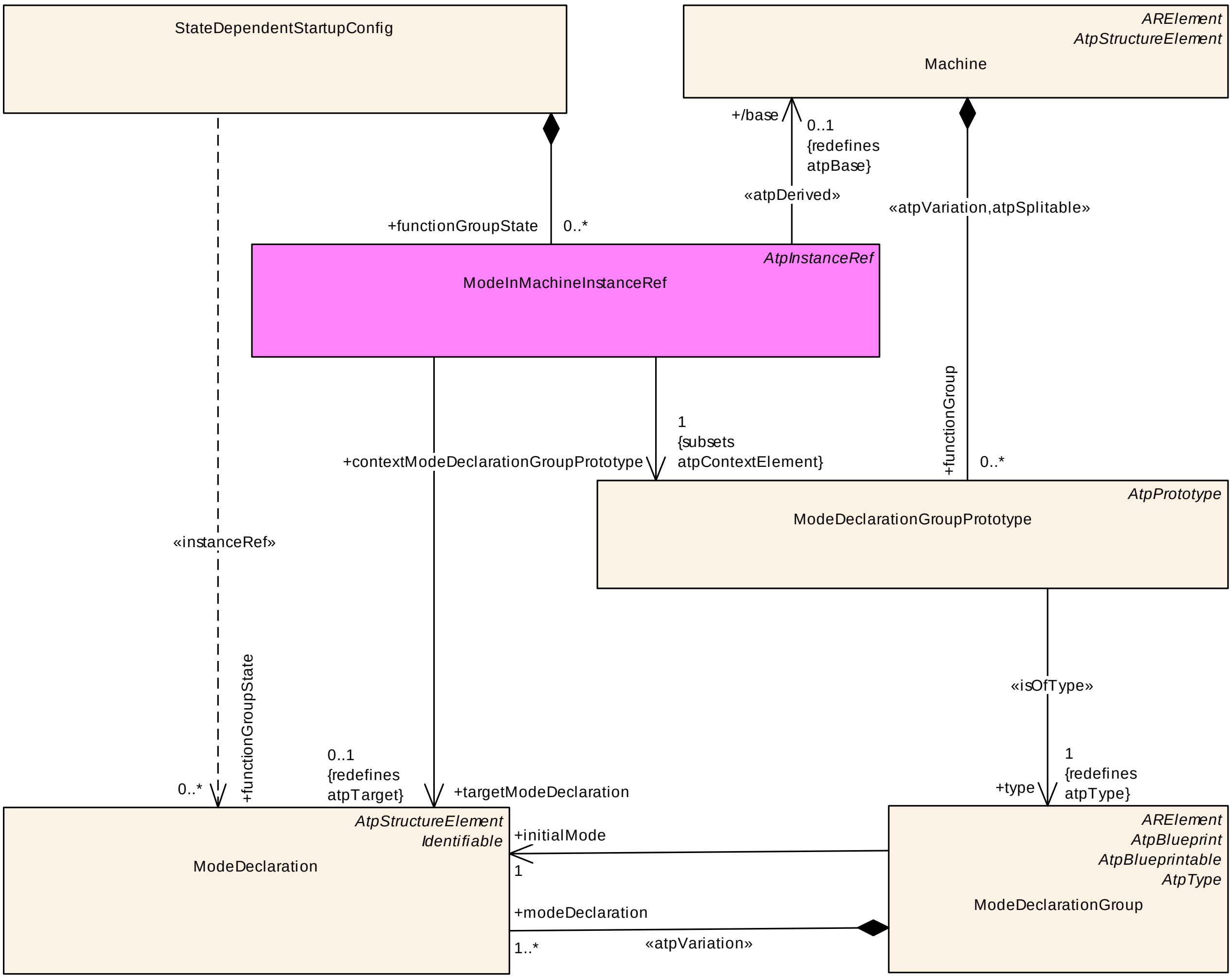


Figure B.13: Modeling of ModeInMachineInstanceRef

Class	**ModeInMachineInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest::InstanceRefs			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Machine	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.14: ModeInMachineInstanceRef

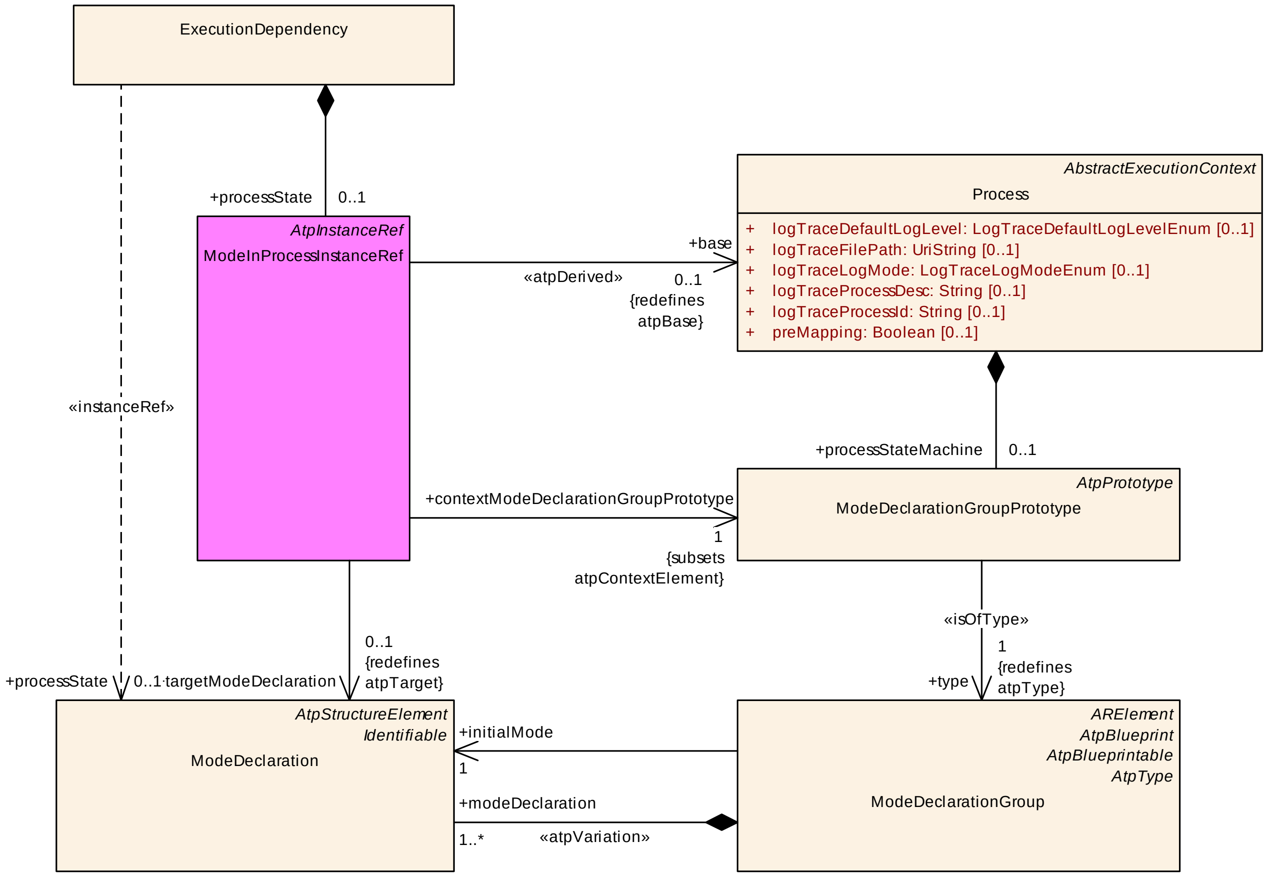


Figure B.14: Modeling of ModeInProcessInstanceRef

Class	**ModeInProcessInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest::InstanceRefs			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
base	Process	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
contextModeDeclarationGroupPrototype	ModeDeclarationGroupPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetModeDeclaration	ModeDeclaration	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.15: ModeInProcessInstanceRef

B.6 Modeling of Diagnostic-related InstanceRefs

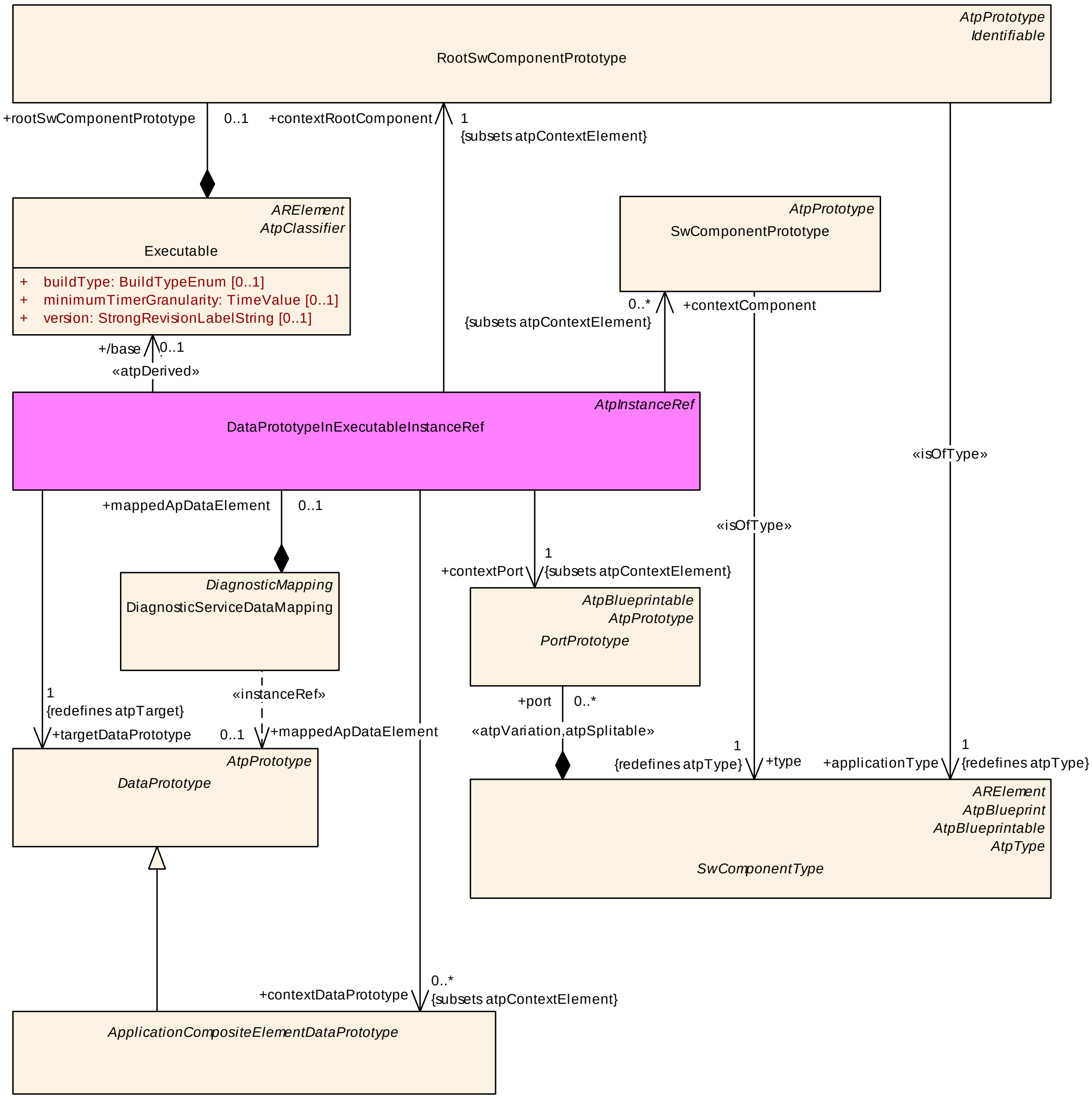


Figure B.15: Modeling of DiagnosticServiceDataMapping via DataPrototypeInExecutableInstanceRef

Class	**DataPrototypeInExecutableInstanceRef**
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping
Note	**Tags:** atp.Status=draft
Base	*ARObject*, *AtpInstanceRef*

▽

△

Class	**DataPrototypeInExecutableInstanceRef**			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
contextComponent	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextDataPrototype	ApplicationCompositeElementDataPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=50
contextPort	PortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40
contextRootComponent	RootSwComponentPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetDataPrototype	DataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=60

Table B.16: DataPrototypeInExecutableInstanceRef

Class	**SwcServiceDependencyInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
contextComponent	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootComponent	RootSwComponentPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetSwcServiceDependency	SwcServiceDependency	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.17: SwcServiceDependencyInExecutableInstanceRef

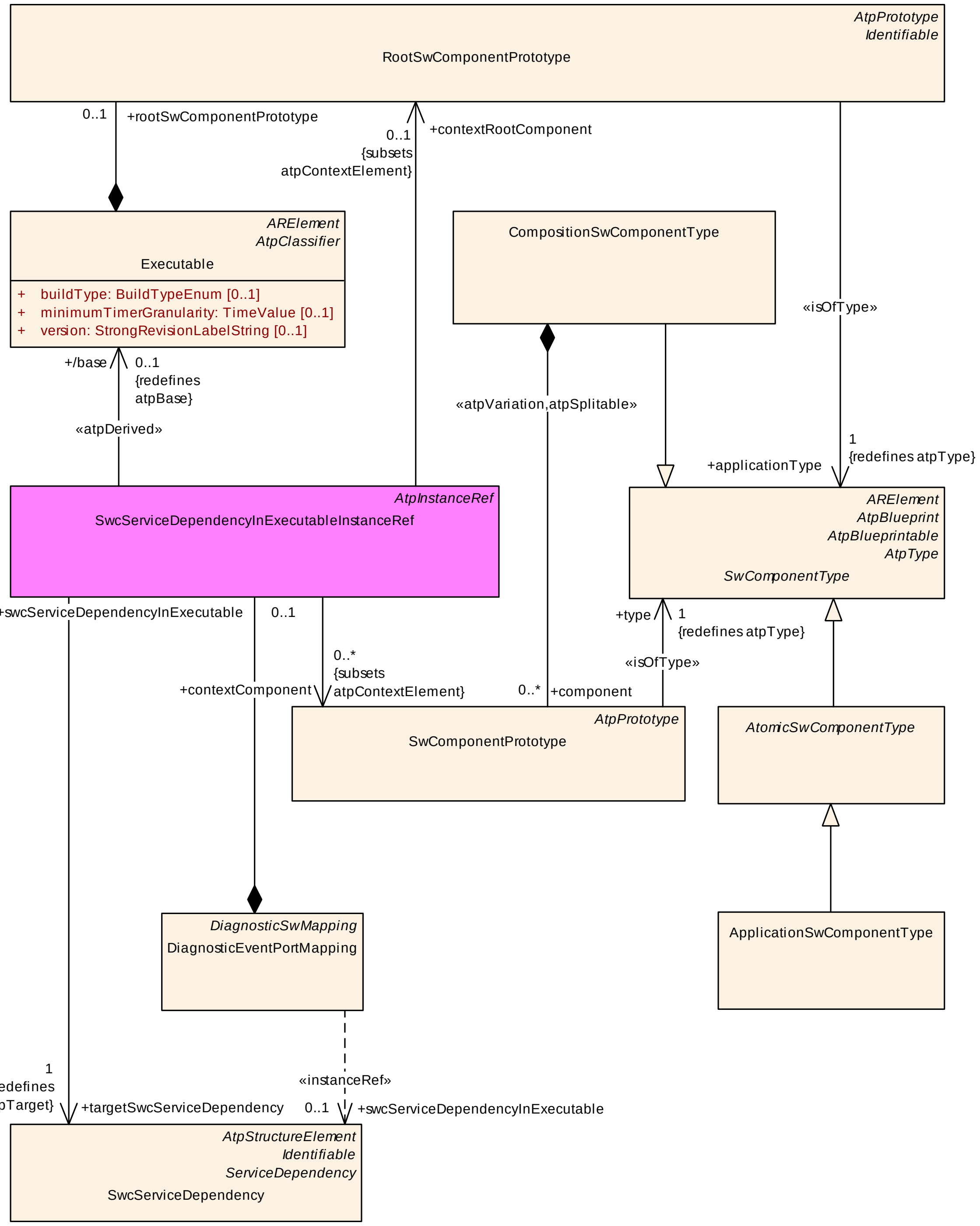


Figure B.16: Modeling of DiagnosticEventPortMapping via SwcServiceDependencyInExecutableInstanceRef

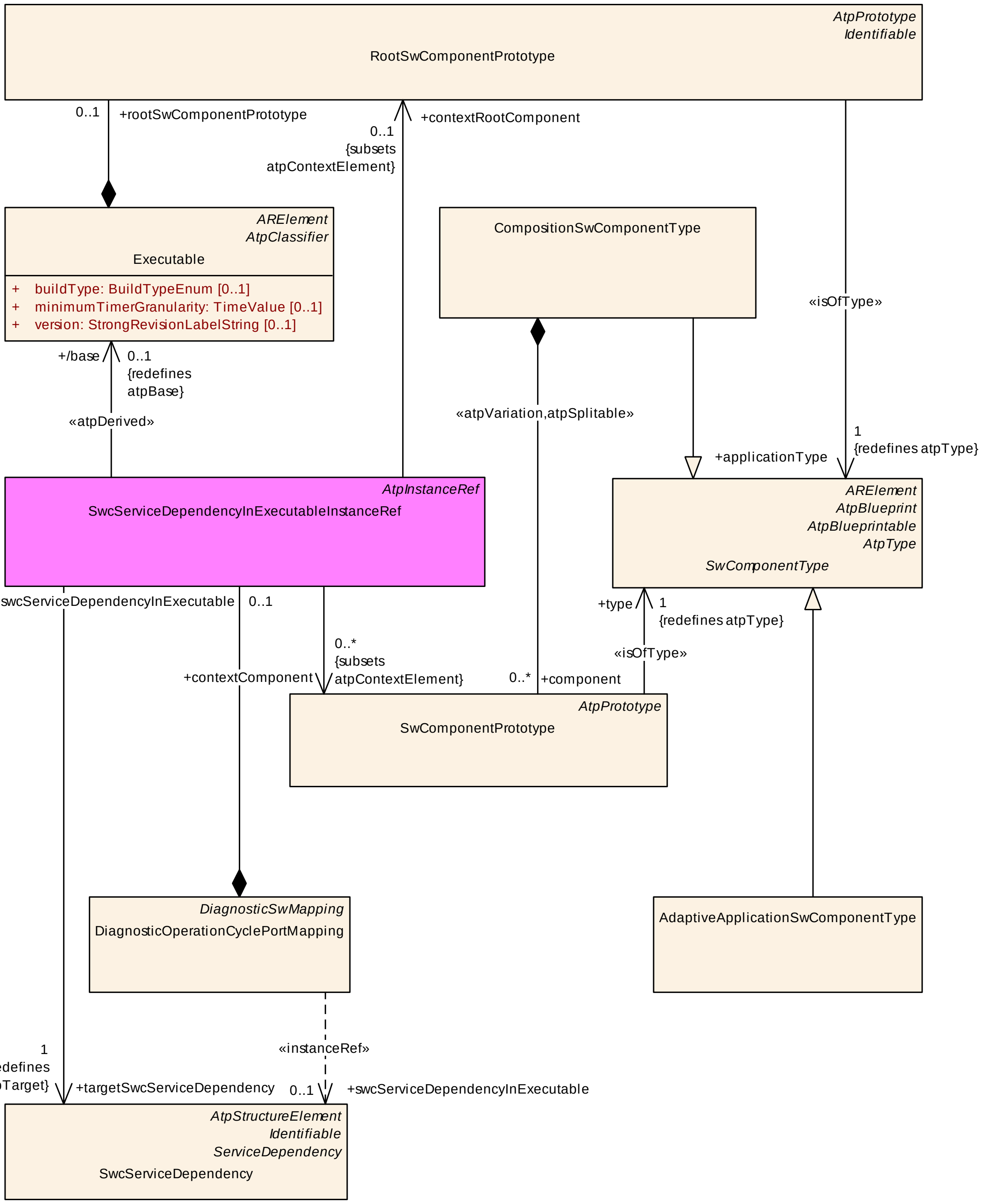


Figure B.17: Modeling of `DiagnosticOperationCyclePortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

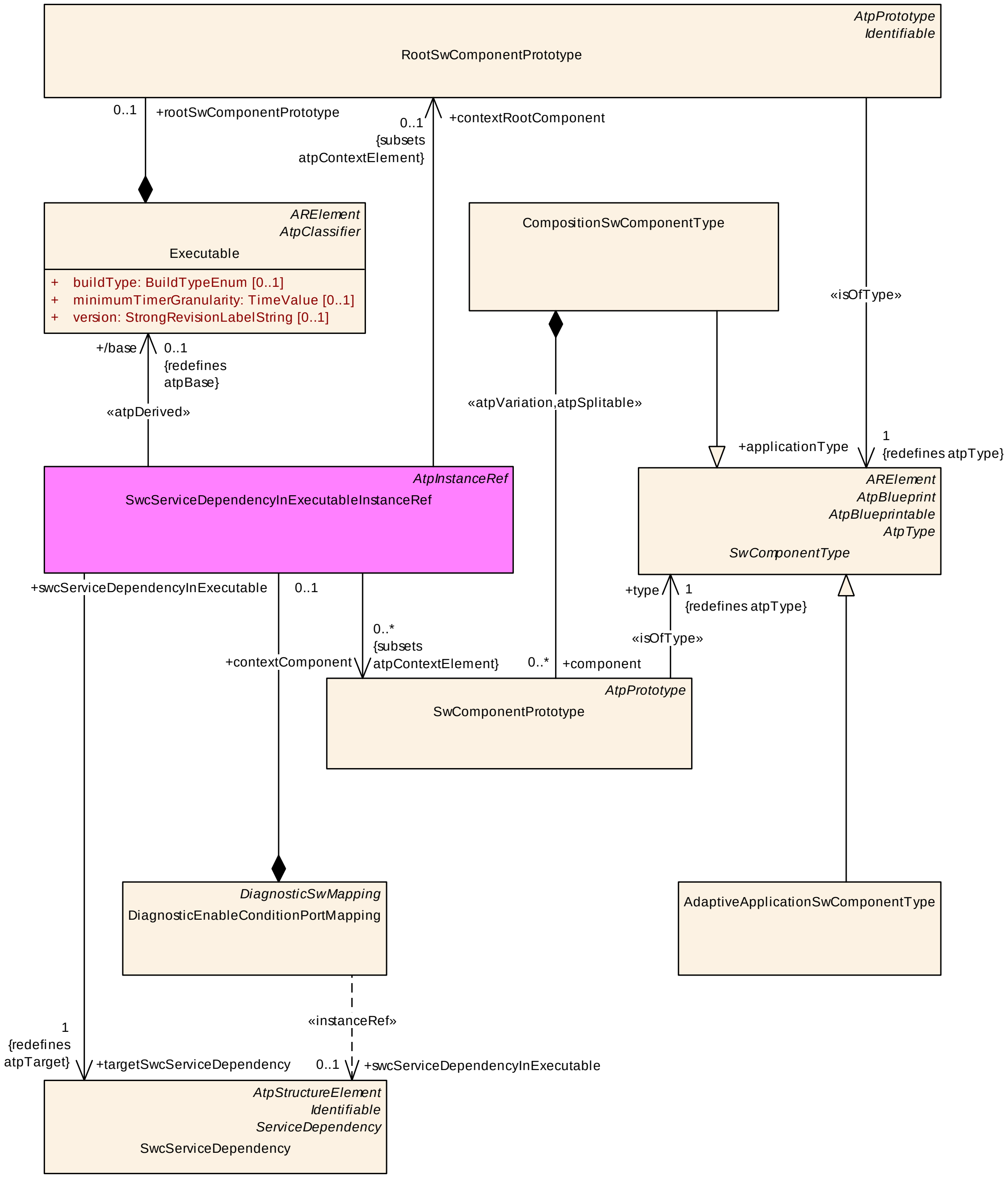


Figure B.18: Modeling of `DiagnosticEnableConditionPortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

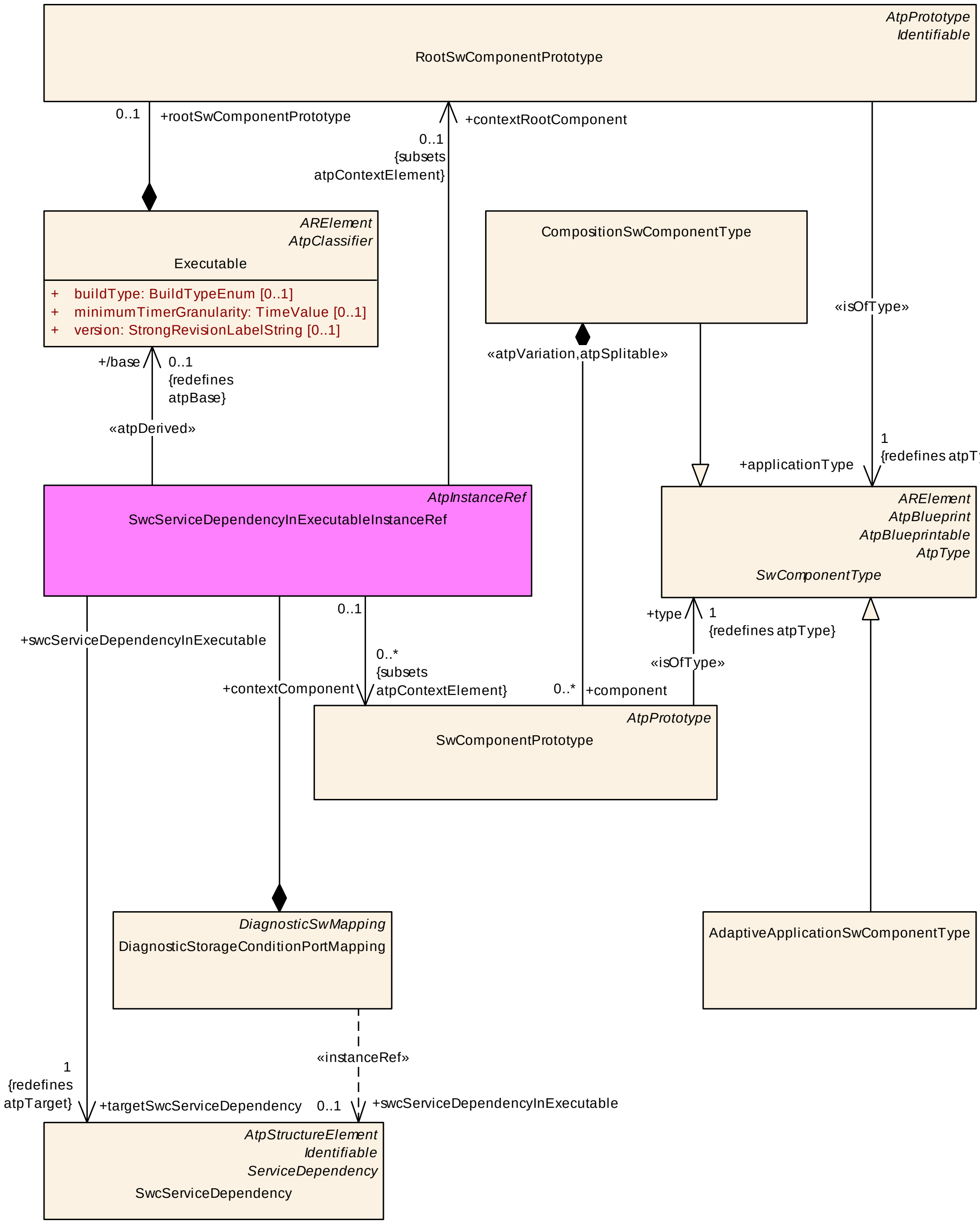


Figure B.19: Modeling of DiagnosticStorageConditionPortMapping via SwcServiceDependencyInExecutableInstanceRef

B.7 Modeling of REST-related InstanceRefs

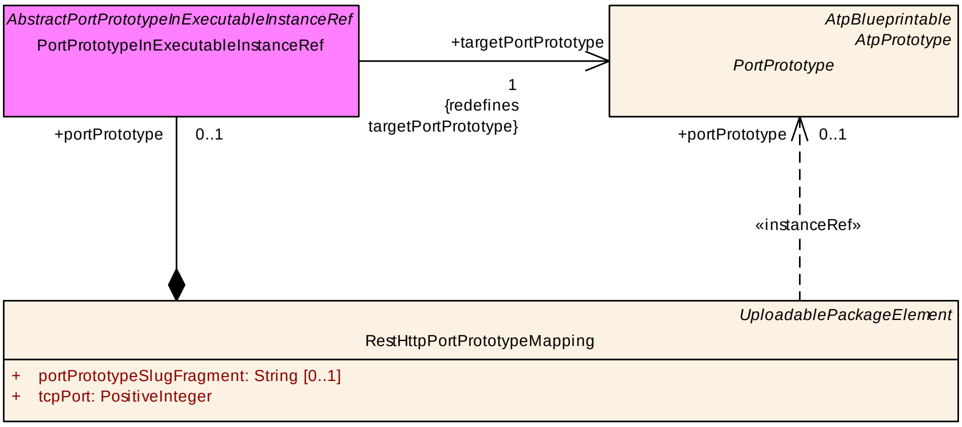


Figure B.20: Modeling of reference RestHttpPortPrototypeMapping.portPrototype

B.8 Modeling of PHM-related InstanceRefs

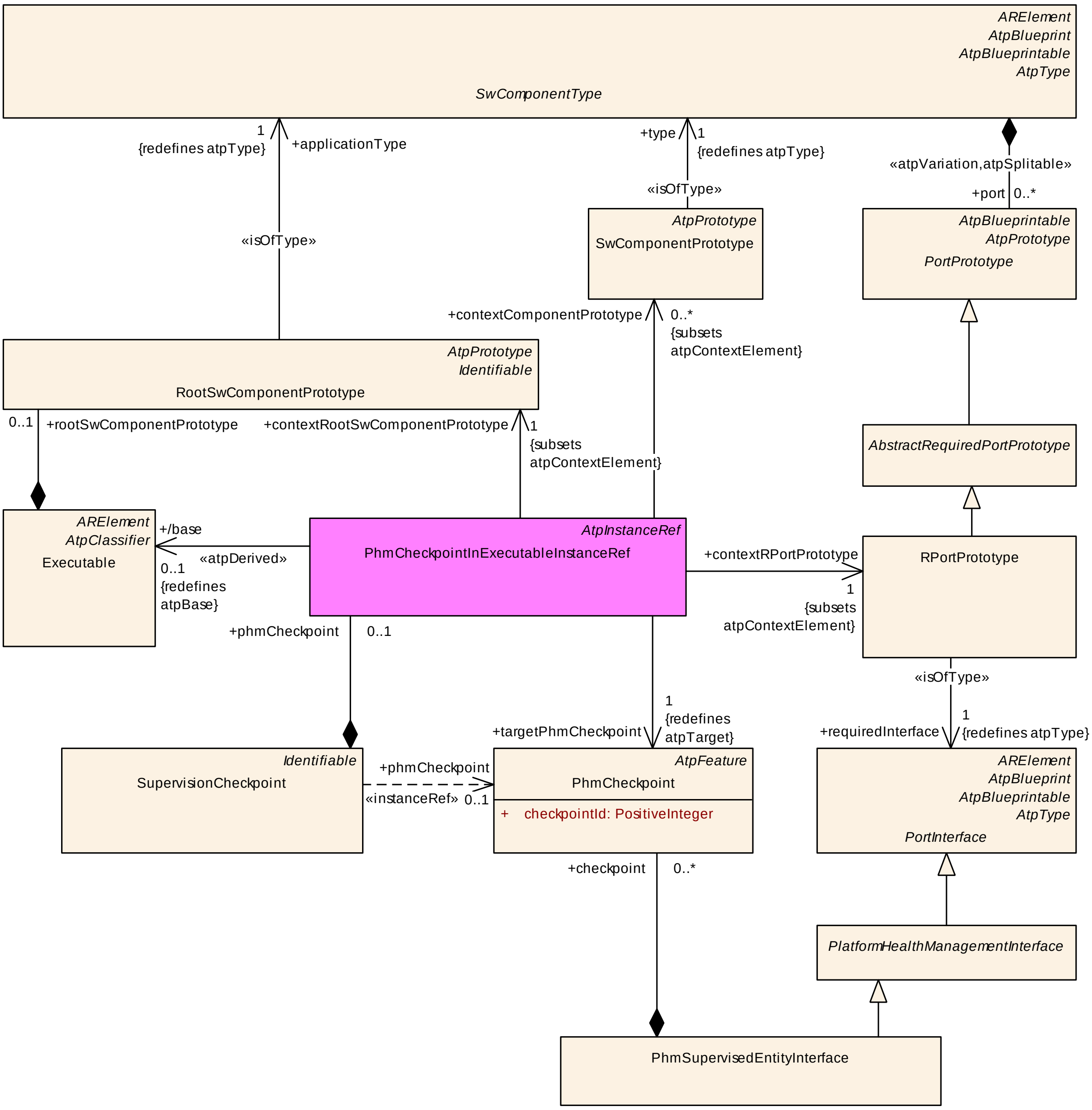


Figure B.21: Modeling of PhmCheckpointInExecutableInstanceRef

Class	**PhmCheckpointInExecutableInstanceRef**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management::InstanceRefs
Note	**Tags:** atp.Status=draft
Base	*ARObject*, *AtpInstanceRef*

▽

△

Class	**PhmCheckpointInExecutableInstanceRef**			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
context Component Prototype	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetPhm Checkpoint	PhmCheckpoint	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=50

Table B.18: PhmCheckpointInExecutableInstanceRef

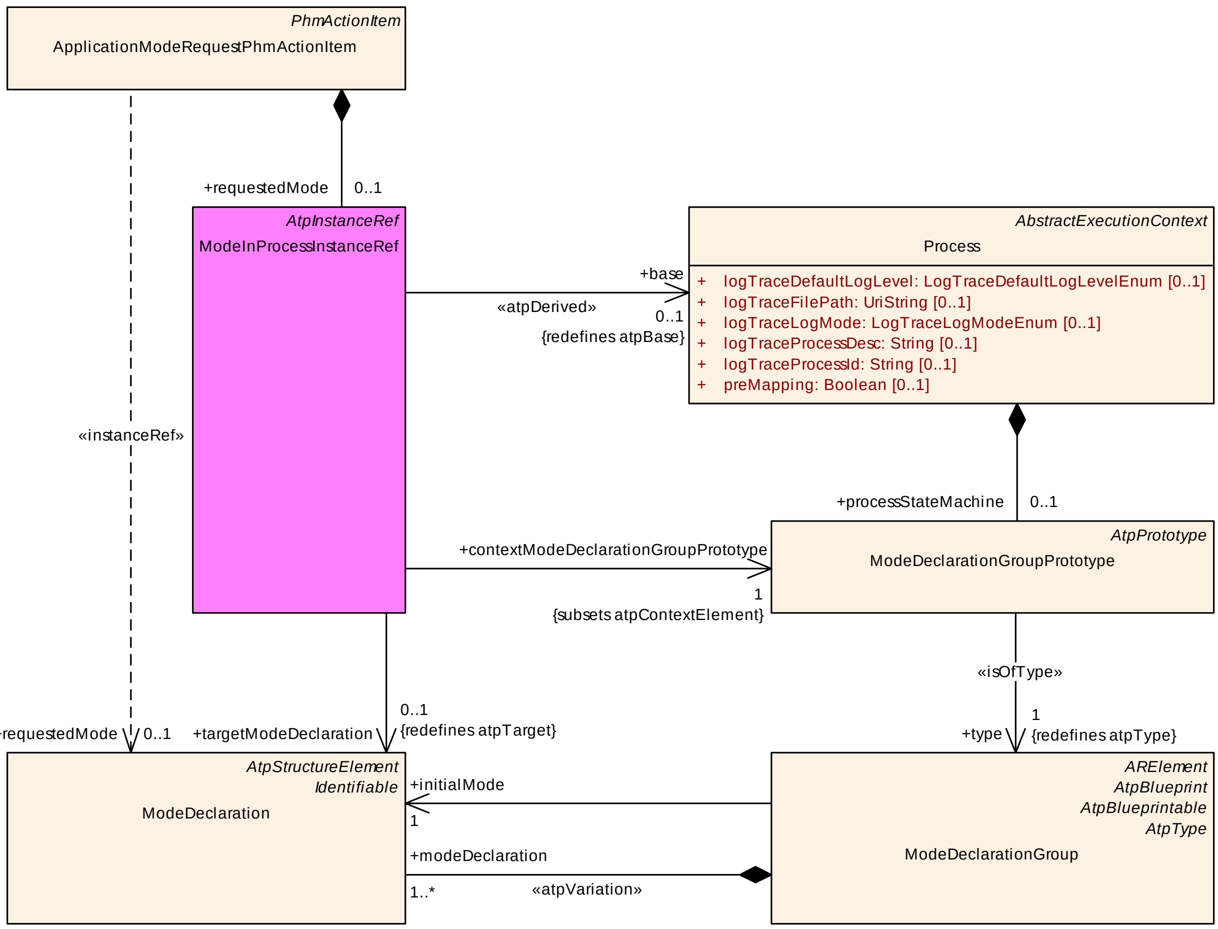


Figure B.22: Modeling of ApplicationModeRequestPhmActionItem.requestedMode

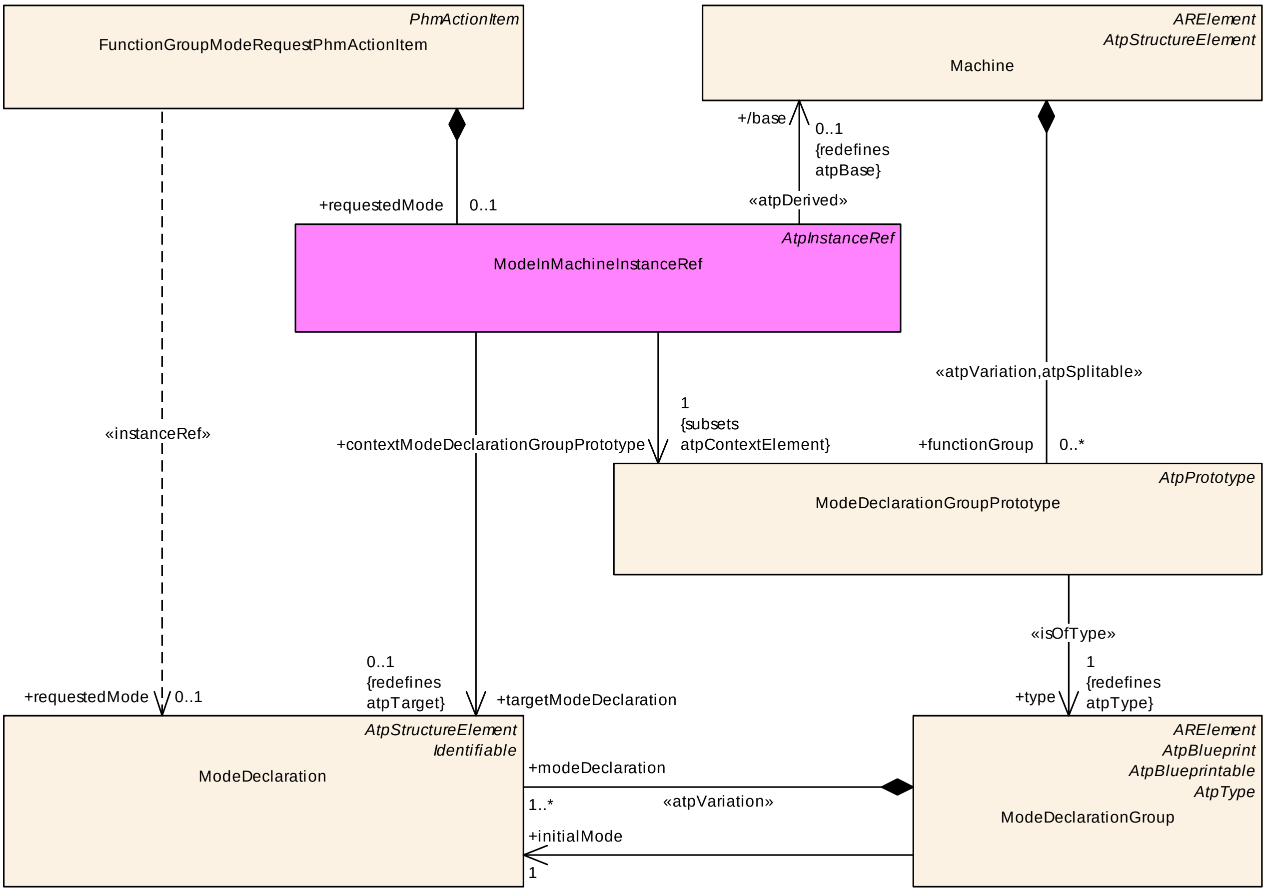


Figure B.23: Modeling of FunctionGroupModeRequestPhmActionItem.requestedMode

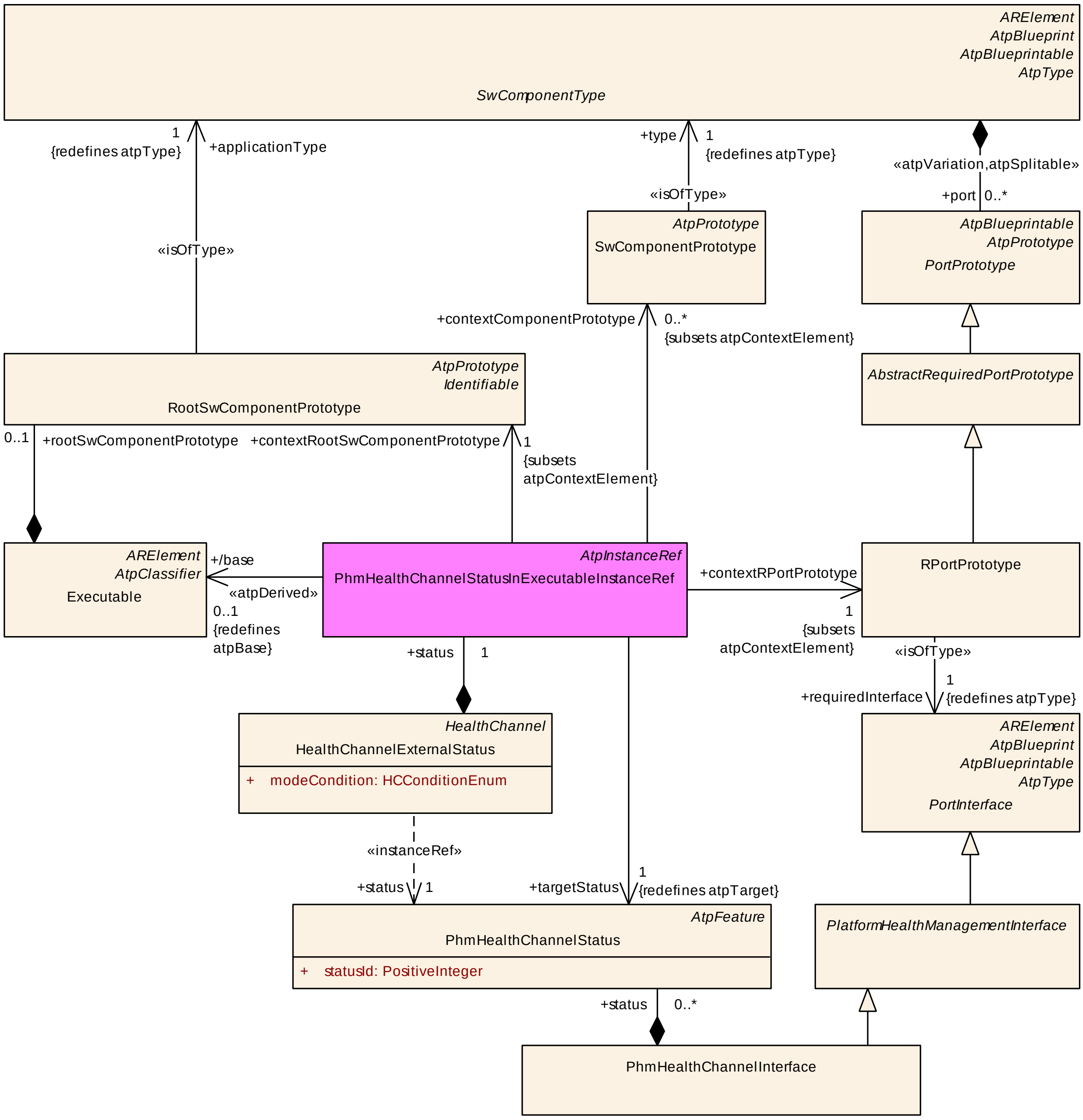


Figure B.24: Modeling of PhmHealthChannelStatusInExecutableInstanceRef

Class	**PhmHealthChannelStatusInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management::InstanceRefs			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10

△

△

Class	PhmHealthChannelStatusInExecutableInstanceRef			
context Component Prototype	SwComponent Prototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRPort Prototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetStatus	PhmHealthChannel Status	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=50

Table B.19: PhmHealthChannelStatusInExecutableInstanceRef

B.9 Modeling of Time-related InstanceRefs

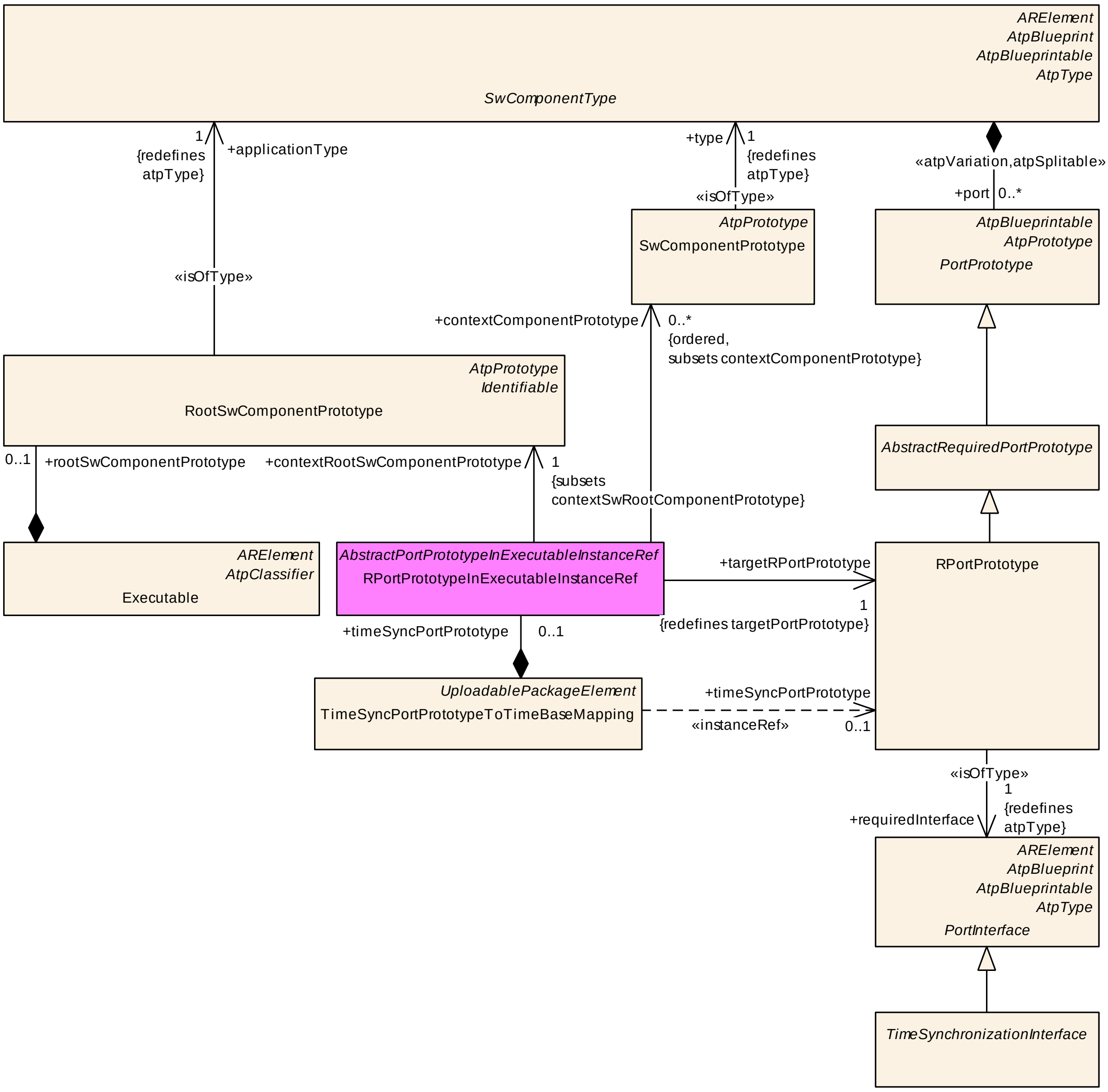


Figure B.25: Modeling of TimeSyncPortPrototypeToTimeBaseMapping.timeSync-PortPrototype

B.10 Modeling of Persistency-related InstanceRefs

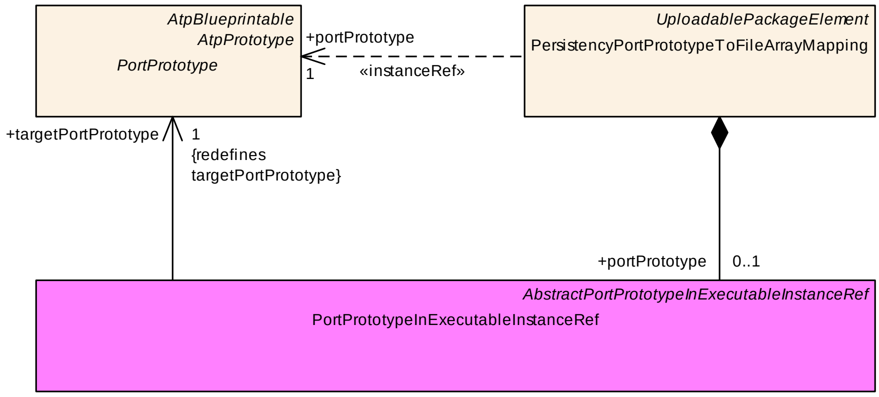


Figure B.26: Modeling of PersistencyPortPrototypeToFileArrayMapping.portPrototype

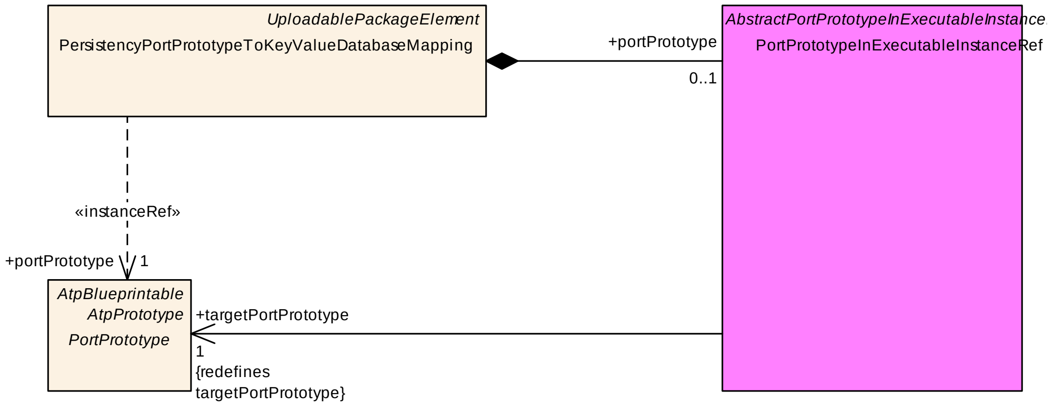


Figure B.27: Modeling of PersistencyPortPrototypeToKeyValueDatabaseMapping.portPrototype

B.11 Modeling of diagnostic-related InstanceRefs

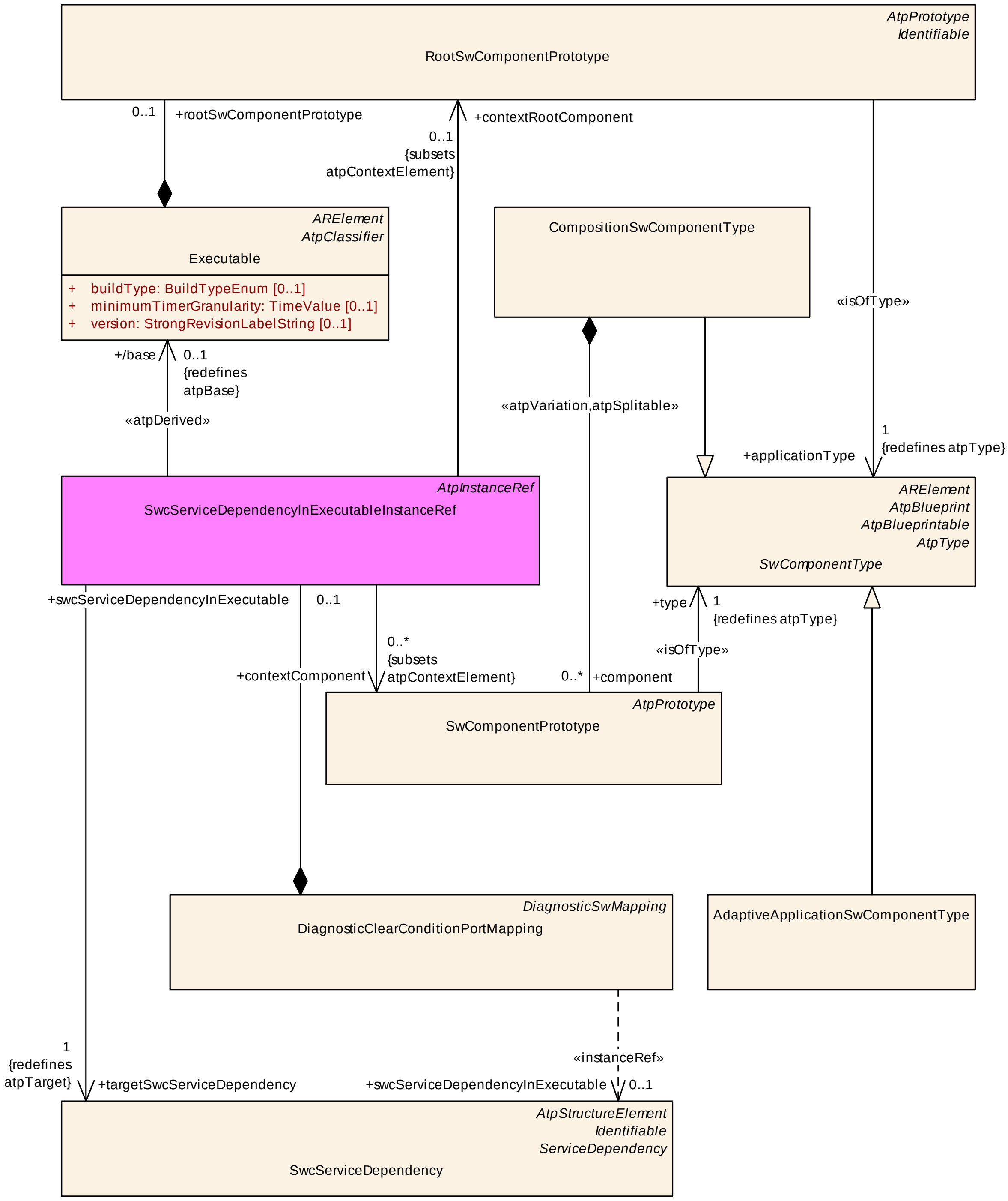


Figure B.28: Modeling of DiagnosticClearConditionPortMapping

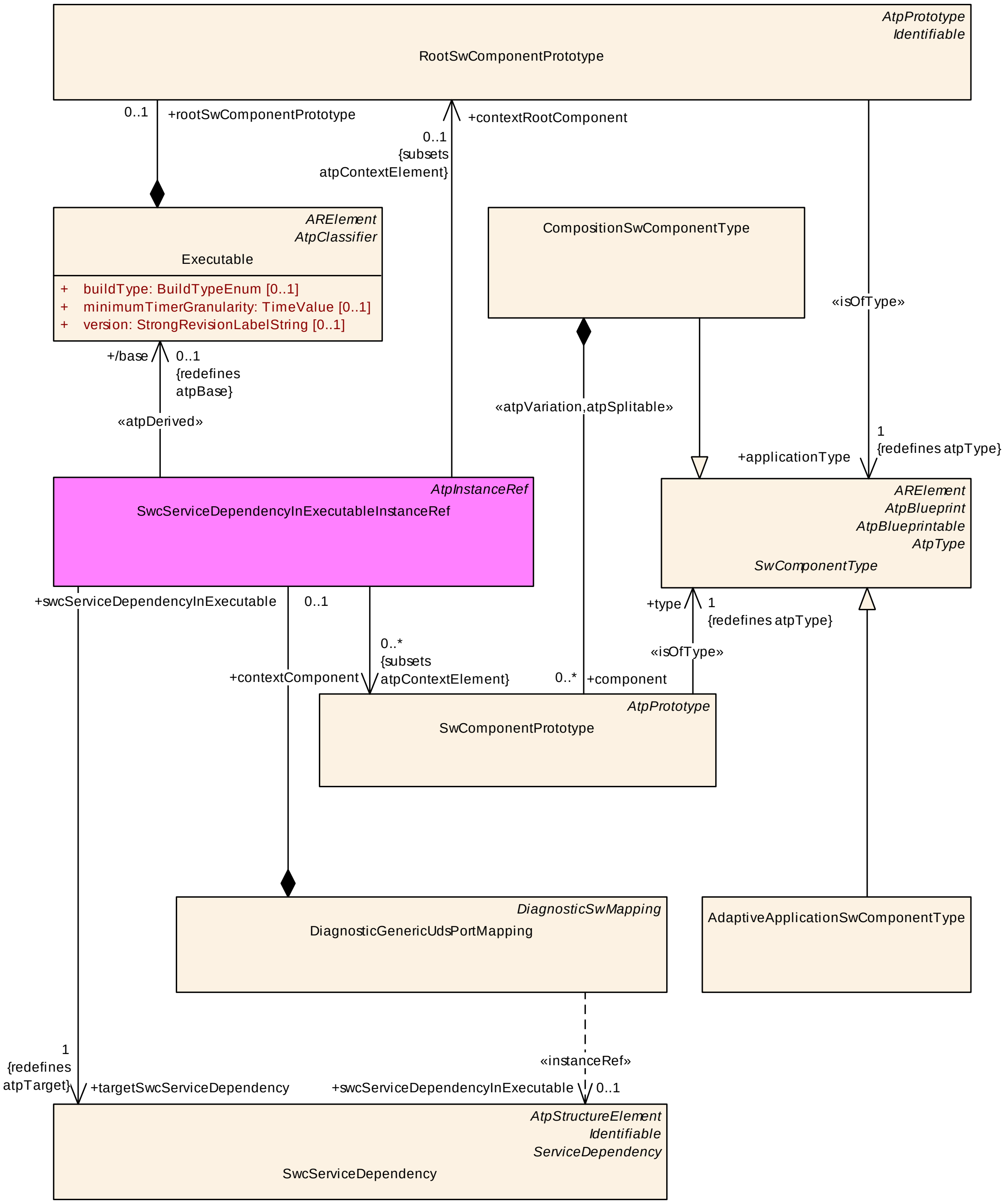


Figure B.29: Modeling of DiagnosticGenericUdsPortMapping

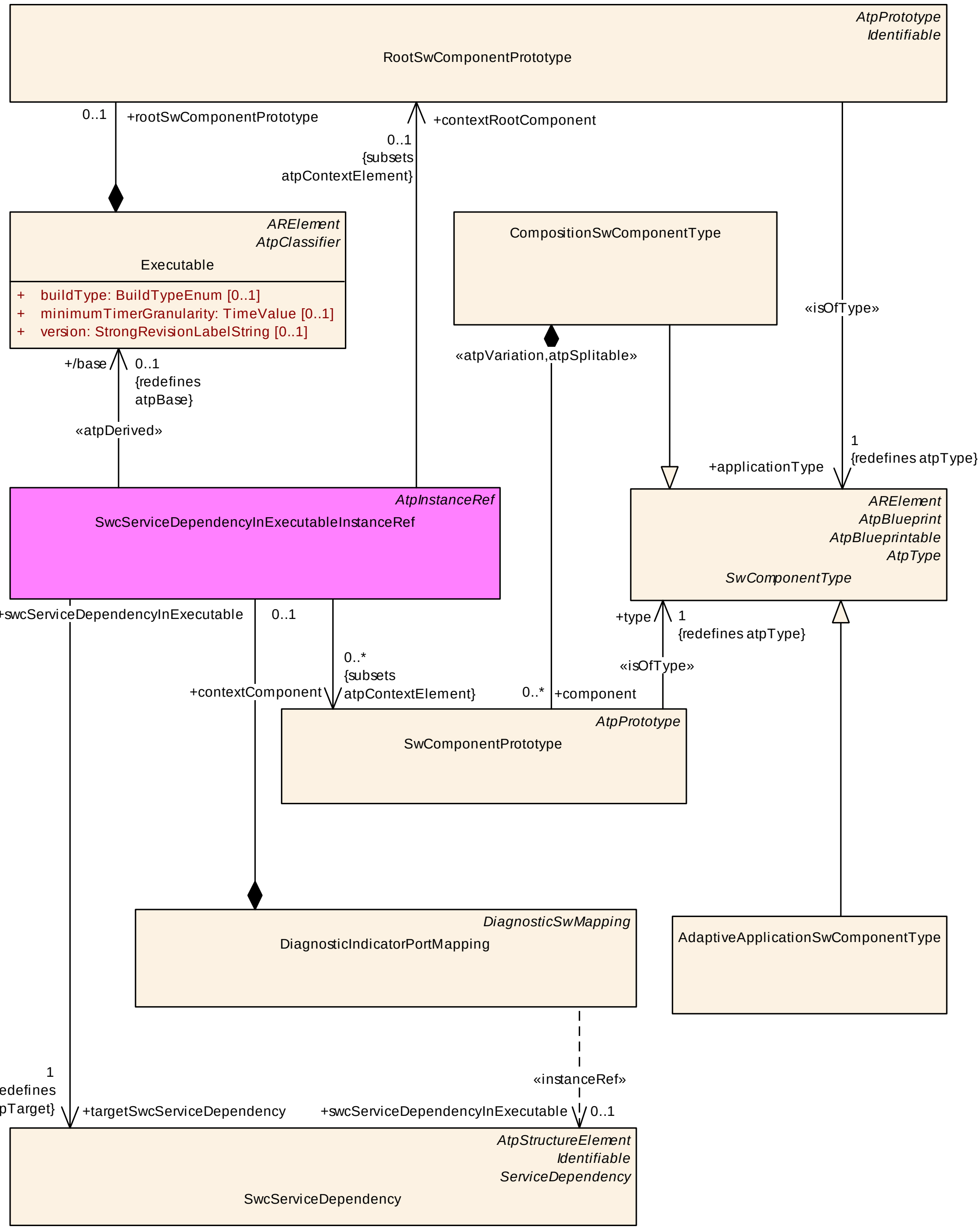


Figure B.30: Modeling of DiagnosticIndicatorPortMapping

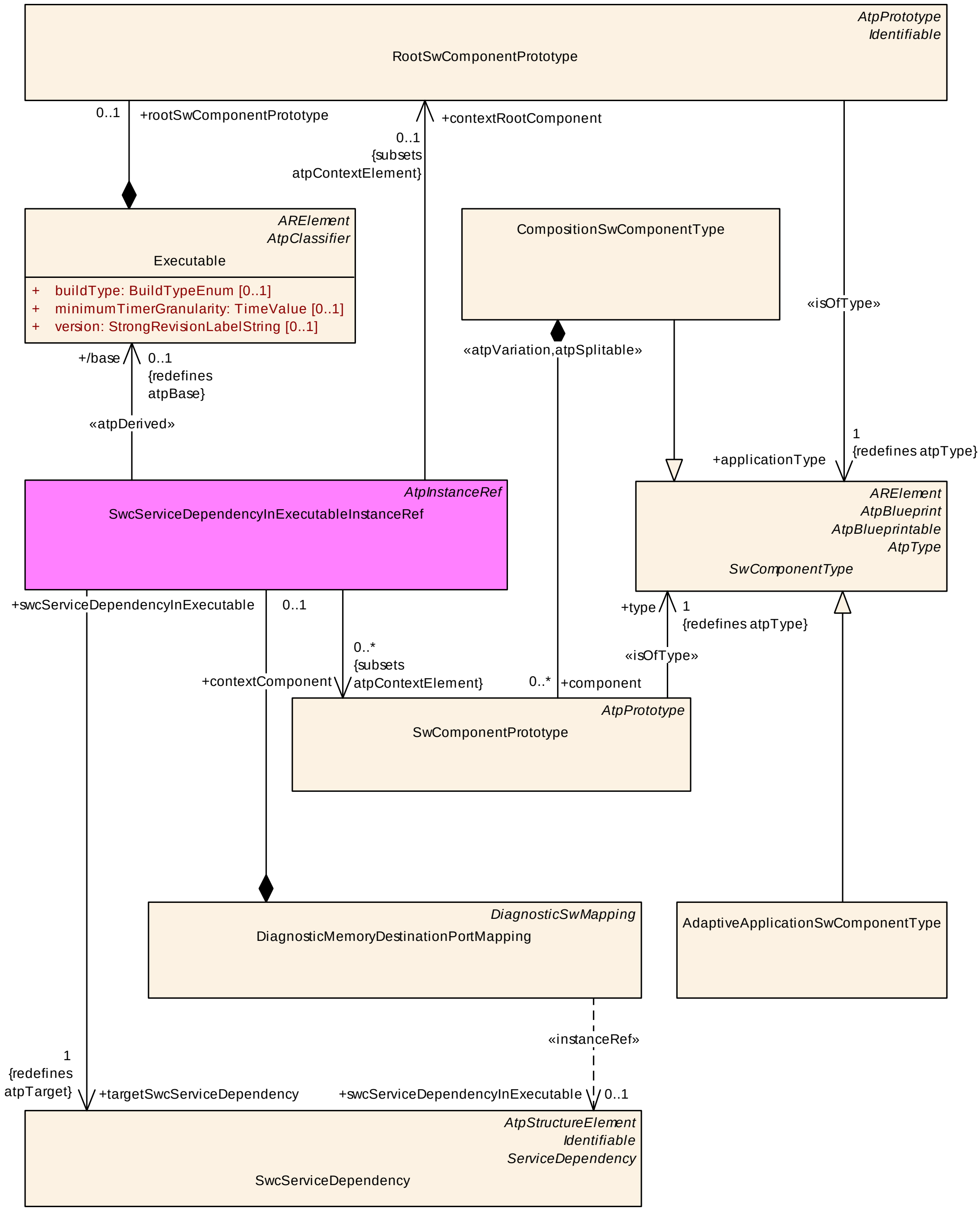


Figure B.31: Modeling of DiagnosticMemoryDestinationPortMapping

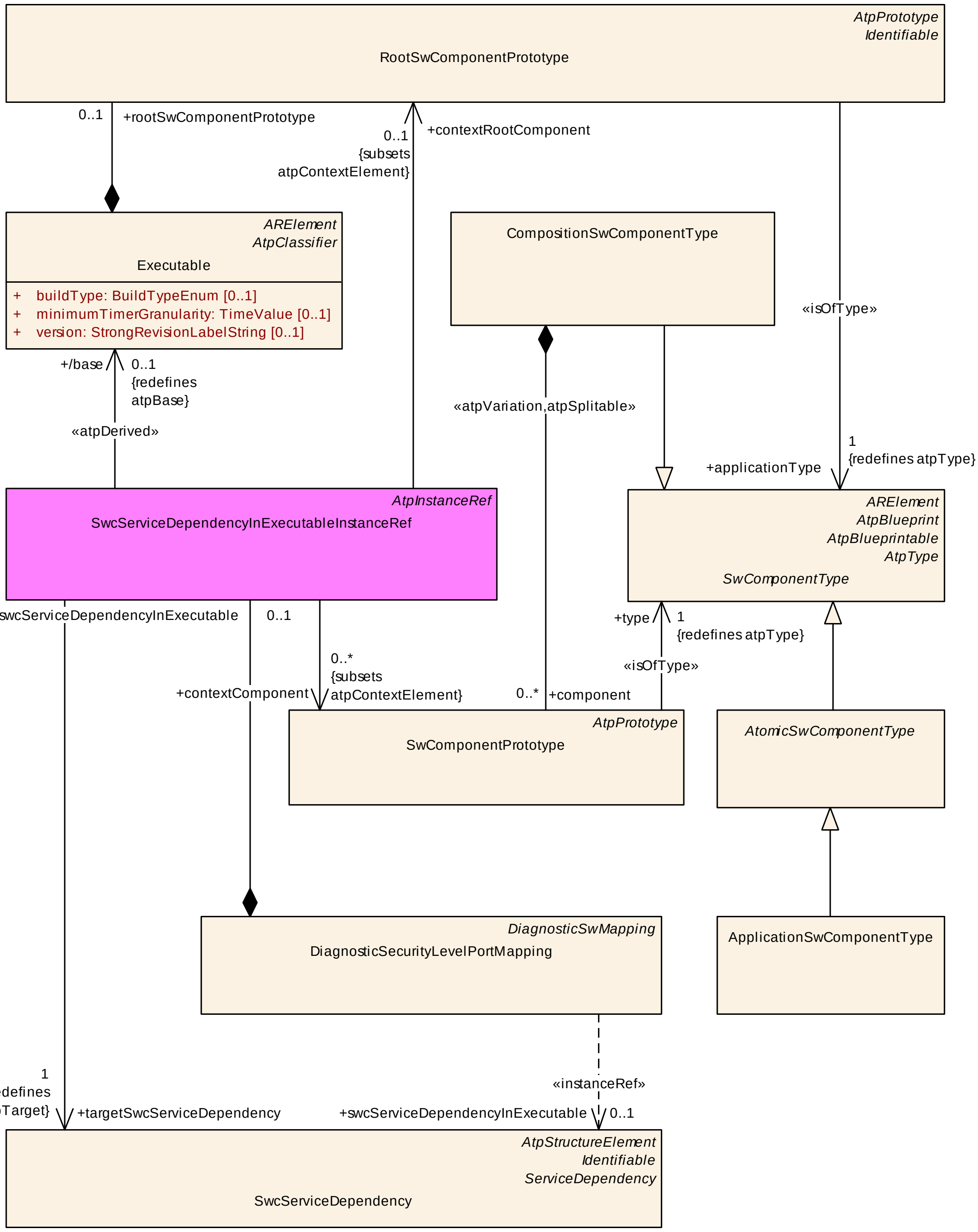


Figure B.32: Modeling of DiagnosticSecurityLevelPortMapping

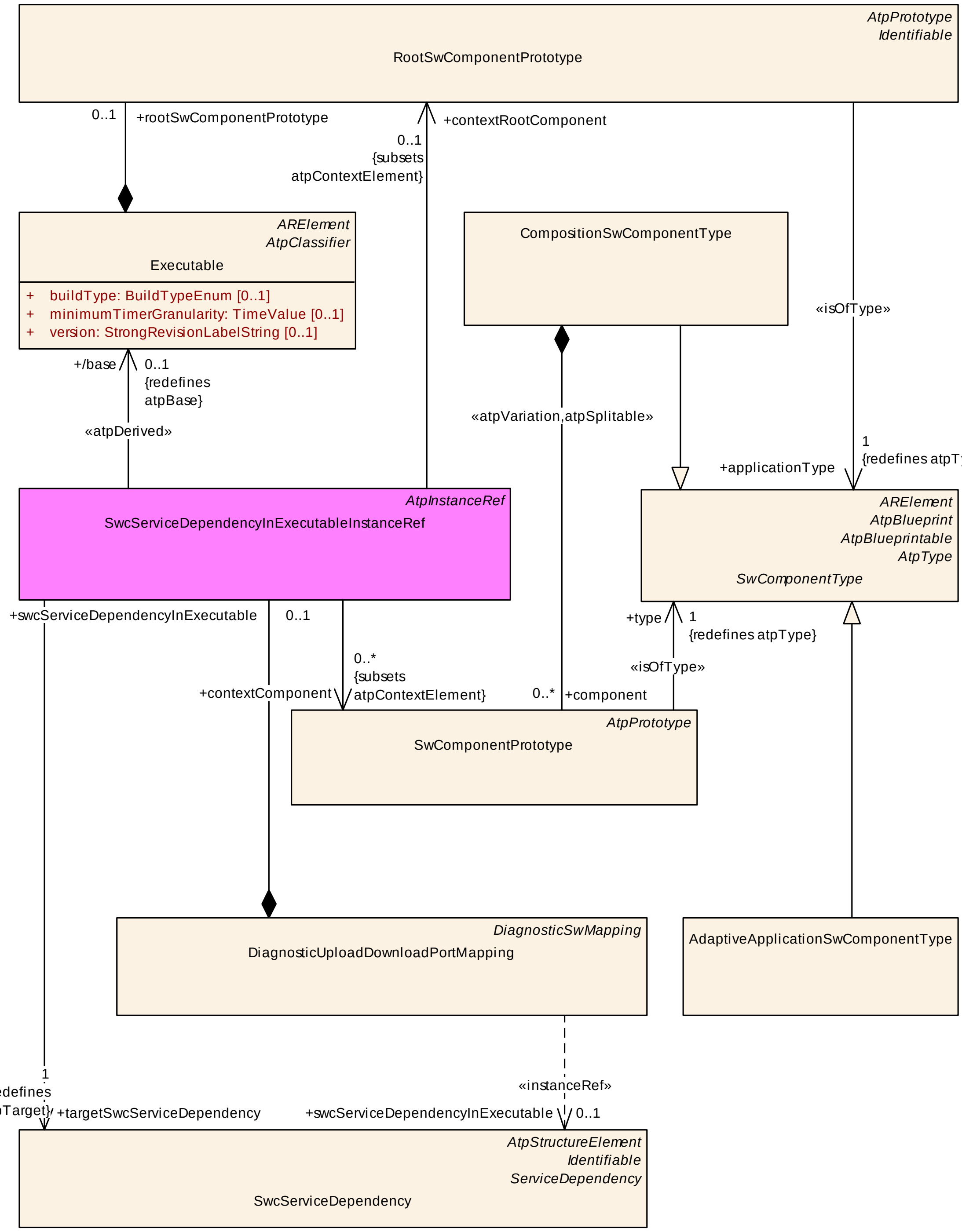


Figure B.33: Modeling of DiagnosticUploadDownloadPortMapping

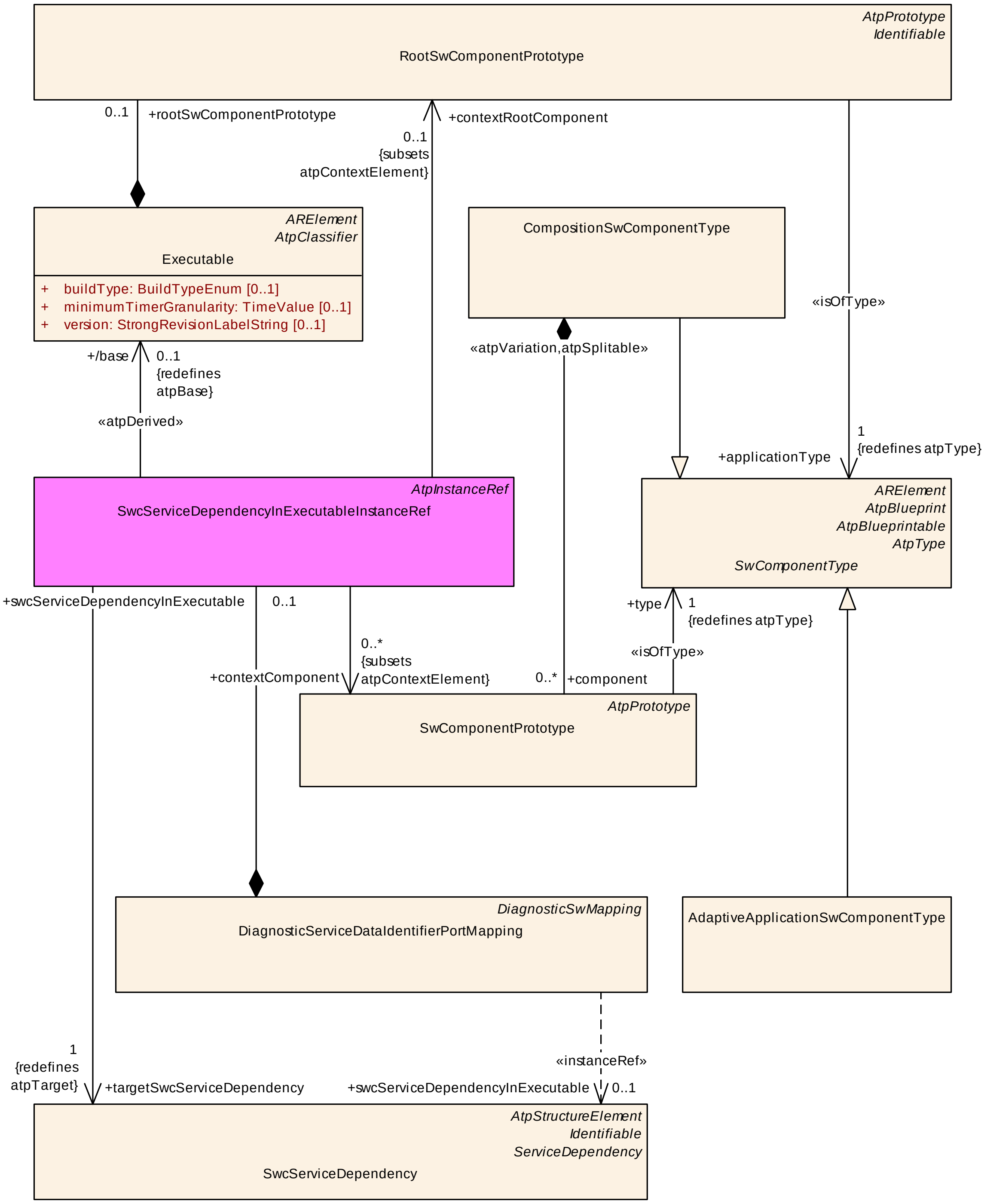


Figure B.34: Modeling of DiagnosticServiceDataIdentifierPortMapping

C Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	***ARElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, Allocator, ApplicationPartition, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BswEntryRelationshipSet, BswModuleDescription, BswModuleEntry, BuildActionManifest, CalibrationParameterValueSet, ClientIdDefinitionSet, ClientServerInterfaceToBswModuleEntryBlueprintMapping, Collection, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CryptoNeedToPortPrototypeMapping, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, DataConstr, DataExchangePoint, DataTransformationSet, DataTypeMappingSet, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DiagnosticMasterToSlaveEventMappingSet, Documentation, EcucDefinitionCollection, EcucDestinationUriDefSet, EcucModuleConfigurationValues, EcucModuleDef, EcucValueCollection, EndToEndProtectionSet, EvaluatedVariantSet, Executable, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FlatMap, GeneralPurposeConnection, *Grant*, *GrantDesign*, HwCategory, HwElement, HwType, IPv6ExtHeaderFilterSet, *Implementation*, InterfaceMappingSet, InterpolationRoutineMappingSet, J1939ControllerApplication, KeywordSet, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, Machine, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, PhmContributionToMachineMapping, PhysicalDimension, PhysicalDimensionMappingSet, *PortInterface*, PortInterfaceMappingSet, PortPrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, ProcessDesign, ProcessDesignToMachineDesignMappingSet, RapidPrototypingScenario, SdgDef, SecureComPropsSet, ServiceInstanceToSignalMappingSet, ServiceInterfaceMappingSet, SoftwareCluster, SoftwareClusterDesign, SoftwarePackage, SomeipDataPrototypeTransformationProps, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValueSet, SwcBswMapping, System, SystemSignal, SystemSignalGroup, TcpOptionFilterSet, *TimingExtension*, TransformationPropsSet, TransformationPropsToServiceInterfaceElementMappingSet, Unit, UnitGroup, *UploadablePackageElement*, ViewMapSet			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.1: ARElement

Class	**ARPackage**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***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=shortName, variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20
referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortLabel xml.sequenceOffset=10

Table C.2: ARPackage

Class	***AbstractProvidedPortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a provided PortPrototype.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Port Prototype*, *Referrable*			
Subclasses	PPortPrototype, PRPortPrototype			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
providedCom Spec	PPortComSpec	*	aggr	Provided communication attributes per interface element (data element or operation).

Table C.3: AbstractProvidedPortPrototype

Class	***AbstractRequiredPortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a required PortPrototype.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Port Prototype*, *Referrable*			
Subclasses	PRPortPrototype, RPortPrototype			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
requiredCom Spec	RPortComSpec	*	aggr	Required communication attributes, one for each interface element.

Table C.4: AbstractRequiredPortPrototype

Class	**AdminData**			
Package	M2::MSR::AsamHdo::AdminData			
Note	AdminData represents the ability to express administrative information for an element. This administration information is to be treated as meta-data such as revision id or state of the file. There are basically four kinds of meta-data • The language and/or used languages. • Revision information covering e.g. revision number, state, release date, changes. Note that this information can be given in general as well as related to a particular company. • Document meta-data specific for a company			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
docRevision (ordered)	DocRevision	*	aggr	This allows to denote information about the current revision of the object. Note that information about previous revisions can also be logged here. The entries shall be sorted descendant by date in order to reflect the history. Therefore the most recent entry representing the current version is denoted first. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=50 xml.typeElement=false xml.typeWrapperElement=false
language	LEnum	0..1	attr	This attribute specifies the master language of the document or the document fragment. The master language is the one in which the document is maintained and from which the other languages are derived from. In particular in case of inconsistencies, the information in the master language is priority. **Tags:** xml.sequenceOffset=20
sdg	Sdg	*	aggr	This property allows to keep special data which is not represented by the standard model. It can be utilized to keep e.g. tool specific data. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=60 xml.typeElement=false xml.typeWrapperElement=false
usedLanguages	MultiLanguagePlainText	0..1	aggr	This property specifies the languages which are provided in the document. Therefore it should only be specified in the top level admin data. For each language provided in the document there is one entry in MultilanguagePlain Text. The content of each entry can be used for illustration of the language. The used language itself depends on the language attribute in the entry. **Tags:** xml.sequenceOffset=30

Table C.5: AdminData

Class	**ApplicationArrayDataType**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes
Note	An application data type which is an array, each element is of the same application data type. **Tags:** atp.recommendedPackage=ApplicationDataTypes

▽

△

Class	ApplicationArrayDataType			
Base	*ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.
element	ApplicationArrayElement	1	aggr	This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine.

Table C.6: ApplicationArrayDataType

Class	ApplicationArrayElement			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Describes the properties of the elements of an application array data type.			
Base	*ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
arraySizeHandling	ArraySizeHandlingEnum	0..1	attr	The way how the size of the array is handled.
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
indexDataType	ApplicationPrimitiveDataType	0..1	ref	This reference can be taken to assign a CompuMethod of category TEXTTABLE to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name.
maxNumberOfElements	PositiveInteger	0..1	attr	The maximum number of elements that the array can contain. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table C.7: ApplicationArrayElement

Class	***ApplicationCompositeDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for all application data types composed of other data types.			
Base	*ARElement, ARObject, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable*			
Subclasses	ApplicationArrayDataType, ApplicationAssocMapDataType, ApplicationRecordDataType			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table C.8: ApplicationCompositeDataType

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

Table C.9: ApplicationError

Class	ApplicationRuleBasedValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents rule based values for DataPrototypes typed by ApplicationDataTypes (ApplicationArrayDataType or a compound ApplicationPrimitiveDataType which also boils down to an array-nature).			
Base	*ARObject*, *AbstractRuleBasedValueSpecification*, *ValueSpecification*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
category	Identifier	1	attr	This represents the category of the RuleBasedValue Specification **Tags:** xml.sequenceOffset=-20
swAxis Cont (ordered)	RuleBasedAxisCont	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second sw AxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	RuleBasedValueCont	0..1	aggr	This represents the values of an array or Compound Primitive Data Type.

Table C.10: ApplicationRuleBasedValueSpecification

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

Table C.11: ApplicationSwComponentType

Class	**ApplicationValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents values for DataPrototypes typed by ApplicationDataTypes (this includes in particular compound primitives). For further details refer to ASAM CDF 2.0. This meta-class corresponds to some extent with SW-INSTANCE in ASAM CDF 2.0.			
Base	*ARObject*, *ValueSpecification*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
category	Identifier	1	attr	Specifies to which category of ApplicationDataType this ApplicationValueSpecification can be applied (e.g. as an initial value), thus imposing constraints on the structure and semantics of the contained values, see [constr_1006] and [constr_2051].
swAxisCont (ordered)	SwAxisCont	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second swAxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	SwValueCont	0..1	aggr	This represents the values of a Compound Primitive Data Type.

Table C.12: ApplicationValueSpecification

Class	**ArgumentDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
direction	ArgumentDirectionEnum	1	attr	This attribute specifies the direction of the argument prototype.
serverArgumentImplPolicy	ServerArgumentImplPolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures.

Table C.13: ArgumentDataPrototype

Enumeration	**ArgumentDirectionEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	Use cases: • Arguments in ClientServerOperation can have different directions that need to be formally indicated because they have an impact on how the function signature looks like eventually. • Arguments in BswModuleEntry already determine a function signature, but the direction is used to specify the semantics, especially of pointer arguments.
Literal	**Description**
in	The argument value is passed to the callee. **Tags:** atp.EnumerationValue=0

▽

△

Enumeration	**ArgumentDirectionEnum**
inout	The argument value is passed to the callee but also passed back from the callee to the caller. **Tags:** atp.EnumerationValue=1
out	The argument value is passed from the callee to the caller. **Tags:** atp.EnumerationValue=2

Table C.14: ArgumentDirectionEnum

Enumeration	**ArraySizeSemanticsEnum**
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes
Note	This type controls how the information about the number of elements in an ApplicationArrayDataType is to be interpreted.
Literal	*Description*
fixedSize	This means that the ApplicationArrayDataType will always have a fixed number of elements. **Tags:** atp.EnumerationValue=0
variableSize	This implies that the actual number of elements in the ApplicationArrayDataType might vary at run-time. The value of arraySize represents the maximum number of elements in the array. **Tags:** atp.EnumerationValue=1

Table C.15: ArraySizeSemanticsEnum

Class	**ArrayValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for an array.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
element (ordered)	ValueSpecification	*	aggr	The value for a single array element. All Value Specifications aggregated by ArrayValueSpecification shall have the same structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table C.16: ArrayValueSpecification

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

Table C.17: AssemblySwConnector

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

Table C.18: AtomicSwComponentType

Class	***AutosarDataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of an AutosarDataType.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ArgumentDataPrototype, Field, ParameterDataPrototype, PersistencyDataElement, VariableDataPrototype			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
type	AutosarDataType	1	tref	This represents the corresponding data type. **Stereotypes:** isOfType

Table C.19: AutosarDataPrototype

Class	***AutosarDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for user defined AUTOSAR data types for ECU software.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractImplementationDataType*, *ApplicationDataType*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType.

Table C.20: AutosarDataType

Class	**BaseTypeDirectDefinition**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	*ARObject*, *BaseTypeDefinition*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
baseType Encoding	BaseTypeEncoding String	1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:** xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:** xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:** xml.sequenceOffset=110
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:** xml.sequenceOffset=100

native Declaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example
				BaseType with
				`shortName: "MyUnsignedInt"`
				`nativeDeclaration: "unsigned short"`
				Results in
				`typedef unsigned short MyUnsignedInt;`
				If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE.
				If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size.
				This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems.
				Tags: xml.sequenceOffset=120

Table C.21: BaseTypeDirectDefinition

Class	**ClientServerInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A client/server interface declares a number of operations that can be invoked on a server by a client. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	ClientServerInterface			
operation	ClientServerOperation	1..*	aggr	ClientServerOperation(s) of this ClientServerInterface. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface.

Table C.22: ClientServerInterface

Class	**CompuConst**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that the value of a computation method scale is constant.			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
compuConst ContentType	CompuConstContent	1	aggr	This is the actual content of the constant compu method scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=10 xml.typeElement=false xml.typeWrapperElement=false

Table C.23: CompuConst

Class	**CompuConstTextContent**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the textual content of a scale.			
Base	*ARObject*, *CompuConstContent*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
vt	VerbatimString	1	attr	This represents a textual constant in the computation method.

Table C.24: CompuConstTextContent

Class	**CompuMethod**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:** atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
compuInternal ToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:** xml.sequenceOffset=80

▽

△

Class	**CompuMethod**			
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:** xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:** xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:** xml.sequenceOffset=30

Table C.25: CompuMethod

Class	**CompuScale**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
desc	MultiLanguageOverviewParagraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:** xml.sequenceOffset=30
compuInverseValue	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. **Tags:** xml.sequenceOffset=60
compuScaleContents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:** xml.sequenceOffset=35
shortLabel	Identifier	0..1	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

▽

△

Class	**CompuScale**			
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 C.26: CompuScale

Class	**ConstantReference**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Instead of defining this value inline, a constant is referenced.			
Base	*ARObject*, *ValueSpecification*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
constant	ConstantSpecification	1	ref	The referenced constant.

Table C.27: ConstantReference

Class	**DataConstr**			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to specify constraints on data. **Tags:** atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table C.28: DataConstr

Class	**DataConstrRule**			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to express one specific data constraint rule.			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*

▽

△

Class	**DataConstrRule**			
constrLevel	Integer	0..1	attr	This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages. The lower the level, the more stringent the check. Used to distinguish hard or soft limits. **Tags:** xml.sequenceOffset=20
internalConstrs	InternalConstrs	0..1	aggr	Describes the limitations applicable on the internal domain (as opposed to the physical domain). **Tags:** xml.sequenceOffset=40
physConstrs	PhysConstrs	0..1	aggr	Describes the limitations applicable on the physical domain (as opposed to the internal domain). **Tags:** xml.sequenceOffset=30

Table C.29: DataConstrRule

Class	***DataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of any data type.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ApplicationCompositeElementDataPrototype*, *AutosarDataPrototype*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
swDataDef Props	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level.

Table C.30: DataPrototype

Class	***DiagnosticAbstractDataIdentifierInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class serves as the abstract base class of PortInterfaces dedicated to the access of diagnostic data identifiers on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticDataElementInterface, DiagnosticDataIdentifierGenericInterface, DiagnosticDataIdentifier Interface			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.31: DiagnosticAbstractDataIdentifierInterface

Class	**DiagnosticContributionSet**
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticContribution

▽

△

Class	DiagnosticContributionSet			
Note	This meta-class represents a root node of a diagnostic extract. It bundles a given set of diagnostic model elements. The granularity of the DiagonsticContributionSet is arbitrary in order to support the aspect of decentralized configuration, i.e. different contributors can come up with an own DiagnosticContribution Set. **Tags:** atp.recommendedPackage=DiagnosticContributionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	Type	Mul.	Kind	Note
common Properties	DiagnosticCommon Props	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire Diagnostic ContributionSet. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=commonProperties
element	DiagnosticCommon Element	*	ref	This represents a DiagnosticCommonElement considered in the context of the DiagnosticContributionSet **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element, variationPoint.shortLabel vh.latestBindingTime=postBuild
serviceTable	DiagnosticServiceTable	*	ref	This represents the collection of DiagnosticServiceTables to be considered in the scope of this Diagnostic ContributionSet. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceTable, variationPoint.short Label vh.latestBindingTime=postBuild

Table C.32: DiagnosticContributionSet

Class	DiagnosticIndicator			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticIndicator			
Note	Definition of an indicator. **Tags:** atp.recommendedPackage=DiagnosticIndicators			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	Type	Mul.	Kind	Note
healingCycle Counter Threshold	PositiveInteger	1	attr	This attribute defines the number of healing cycles for the WarningIndicatorOffCriteria **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
type	DiagnosticIndicatorType Enum	0..1	attr	Defines the type of the indicator.

Table C.33: DiagnosticIndicator

Class	***DiagnosticMapping*** (abstract)
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping
Note	Abstract element for different kinds of diagnostic mappings.
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*

▽

△

Class	***DiagnosticMapping*** (abstract)			
Subclasses	DiagnosticEventToDebounceAlgorithmMapping, DiagnosticEventToEnableConditionGroupMapping, DiagnosticEventToOperationCycleMapping, DiagnosticEventToStorageConditionGroupMapping, DiagnosticEventToTroubleCodeJ1939Mapping, DiagnosticEventToTroubleCodeUdsMapping, DiagnosticFimAliasEventGroupMapping, DiagnosticFimAliasEventMapping, DiagnosticInhibitSourceEventMapping, DiagnosticJ1939SpnMapping, DiagnosticProvidedDataMapping, DiagnosticServiceDataMapping, *DiagnosticSwMapping*, DiagnosticTroubleCodeUdsToClearConditionGroupMapping, DiagnosticTroubleCodeUdsToTroubleCodeObdMapping			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.34: DiagnosticMapping

Class	***DiagnosticMemoryDestination*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This abstract meta-class represents a possible memory destination for a diagnostic event.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticMemoryDestinationMirror, DiagnosticMemoryDestinationPrimary, DiagnosticMemoryDestinationUserDefined			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.35: DiagnosticMemoryDestination

Class	**DiagnosticSecurityLevel**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This meta-class represents the ability to define a security level considered for diagnostic purposes. **Tags:** atp.recommendedPackage=DiagnosticSecurityLevels			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
accessDataRecordSize	PositiveInteger	0..1	attr	This represents the size of the AccessDataRecord used in GetSeed. Unit:byte.
keySize	PositiveInteger	1	attr	This represents the size of the security key. Unit: byte.
numFailedSecurityAccess	PositiveInteger	0..1	attr	This represents the number of failed security accesses after which the delay time is activated.
securityDelayTime	TimeValue	1	attr	This represents the delay time after a failed security access. Unit: second.
seedSize	PositiveInteger	1	attr	This represents the size of the security seed. Unit: byte.

Table C.36: DiagnosticSecurityLevel

Class	***DiagnosticServiceInstance*** (abstract)
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService
Note	This represents a concrete instance of a diagnostic service.
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	***DiagnosticServiceInstance*** (abstract)			
Subclasses	DiagnosticClearDiagnosticInformation, DiagnosticClearResetEmissionRelatedInfo, DiagnosticComControl, DiagnosticControlDTCSetting, DiagnosticCustomServiceInstance, *DiagnosticDataByIdentifier*, DiagnosticDynamicallyDefineDataIdentifier, DiagnosticEcuReset, DiagnosticIOControl, *DiagnosticMemoryByAddress*, DiagnosticReadDTCInformation, DiagnosticReadDataByPeriodicID, DiagnosticRequestControlOfOnBoardDevice, DiagnosticRequestCurrentPowertrainData, DiagnosticRequestEmissionRelatedDTC, DiagnosticRequestEmissionRelatedDTCPermanentStatus, DiagnosticRequestFileTransfer, DiagnosticRequestOnBoardMonitoringTestResults, DiagnosticRequestPowertrainFreezeFrameData, DiagnosticRequestVehicleInfo, DiagnosticResponseOnEvent, DiagnosticRoutineControl, DiagnosticSecurityAccess, DiagnosticSessionControl			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
accessPermission	DiagnosticAccessPermission	0..1	ref	This represents the collection of DiagnosticAccessPermissions that allow for the execution of the referencing DiagnosticServiceInstance..
serviceClass	DiagnosticServiceClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticServiceInstance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. **Stereotypes:** atpAbstract

Table C.37: DiagnosticServiceInstance

Class	**DiagnosticServiceSwMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::ServiceMapping			
Note	This represents the ability to define a mapping of a diagnostic service to a software-component or a basic-software module. If the former is used then this kind of service mapping is applicable for the usage of ClientServerInterfaces. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
diagnosticDataElement	DiagnosticDataElement	0..1	ref	This represents a DiagnosticDataElement required to execute the respective diagnostic service in the context of the diagnostic service mapping,
mappedBswServiceDependency	BswServiceDependencyIdent	0..1	ref	This is supposed to represent a reference to a BswServiceDependency. the latter is not derived from Referrable and therefore this detour needs to be implemented to still let BswServiceDependency become the target of a reference.
mappedFlatSwcServiceDependency	SwcServiceDependency	0..1	ref	This represents the ability to refer to an AtomicSwComponentType that is available without the definition of how it will be embedded into the component hierarchy.
mappedSwcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This represents the ability to point into the component hiearchy of an adaptive AUTOSAR model (under possible consideration of the rootSoftwareComposition) **Tags:** atp.Status=draft
mappedSwcServiceDependencyInSystem	SwcServiceDependency	0..1	iref	This represents the ability to point into the component hiearchy (under possible consideration of the rootSoftwareComposition)

▽

△

Class	DiagnosticServiceSwMapping			
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=draft
serviceInstance	DiagnosticService Instance	0..1	ref	This represents the service instance that needs to be considered in this diagnostics service mapping.

Table C.38: DiagnosticServiceSwMapping

Class	**DiagnosticTroubleCodeUds**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This element is used to describe non OBD-relevant DTCs. **Tags:** atp.recommendedPackage=DiagnosticTroubleCodes			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticTroubleCode*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
considerPto Status	Boolean	0..1	attr	This attribute describes the affection of the event by the Dem PTO handling. True: the event is affected by the Dem PTO handling. False: the event is not affected by the Dem PTO handling.
dtcProps	DiagnosticTroubleCode Props	0..1	ref	Defined properties associated with the DemDTC.
eventObd Readiness Group	NameToken	0..1	attr	This attribute specifies the Event OBD Readiness group for PID $01 and PID $41 computation. This attribute is only applicable for emission-related ECUs.
functionalUnit	PositiveInteger	0..1	attr	This attribute specifies a 1-byte value which identifies the corresponding basic vehicle / system function which reports the DTC. This parameter is necessary for the report of severity information.
severity	DiagnosticUdsSeverity Enum	0..1	attr	DTC severity according to ISO 14229-1.
udsDtcValue	PositiveInteger	0..1	attr	Unique Diagnostic Trouble Code value for UDS.
wwhObdDtc Class	DiagnosticWwhObdDtc ClassEnum	0..1	attr	This attribute is used to identify (if applicable) the corresponding severity class of an WWH-OBD DTC.

Table C.39: DiagnosticTroubleCodeUds

Class	**EcuInstance**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. **Tags:** atp.recommendedPackage=EcuInstances			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	**EcuInstance**			
associatedComIPduGroup	ISignalIPduGroup	*	ref	With this reference it is possible to identify which ISignalIPduGroups are applicable for which Communication Connector/ ECU. Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPduGroups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an EcuInstance via the top level ISignalIPduGroup.
associatedPdurIPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PduRIPdu Groups are applicable for which Communication Connector/ ECU.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication.
comConfigurationGwTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRouteSignals of the AUTOSAR COM module in seconds.
comConfigurationRxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRx of the AUTOSAR COM module in seconds.
comConfigurationTxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionTx of the AUTOSAR COM module in seconds.
comEnableMDTForCyclicTransmission	Boolean	0..1	attr	Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclicTiming assigned or eventControlledTiming with numberOfRepetitions > 0).
commController	CommunicationController	1..*	aggr	CommunicationControllers of the ECU.
connector	Communication Connector	*	aggr	All channels controlled by a single controller.
diagnosticAddress	Integer	0..1	attr	An ECU specific ID for responses of diagnostic routines.
ethSwitchPortGroupDerivation	Boolean	0..1	attr	Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done.
partition	EcuPartition	*	aggr	Optional definition of Partitions within an Ecu.
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.
pncPrepareSleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
sleepModeSupported	Boolean	1	attr	Specifies whether the ECU instance may be put to a "low power mode" • true: sleep mode is supported • false: sleep mode is not supported Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software.
v2xSupported	V2xSupportEnum	0..1	attr	This attribute is used to control the existence of the V2X stack on the given EcuInstance.

▽

△

Class	**EcuInstance**			
wakeUpOverBusSupported	Boolean	1	attr	Driver support for wakeup over Bus.

Table C.40: EcuInstance

Class	**EthernetNetworkConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class defines the attributes for the configuration of a port, protocol type and IP address of the communication on a VLAN. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *NetworkConfiguration*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
communicationConnector	EthernetCommunicationConnector	0..1	ref	Reference to the CommunicationConnector (VLAN) for which the network configuration is defined. **Tags:** atp.Status=draft
ipv4MulticastIpAddress	Ip4AddressString	0..1	attr	Multicast IPv4 Address to which the message will be transmitted.
ipv6MulticastIpAddress	Ip6AddressString	0..1	attr	Multicast IPv6 Address to which the message will be transmitted.
tcpPort	PositiveInteger	0..1	attr	This attribute allows to configure a tcp port number.
udpNmCluster	UdpNmCluster	0..1	ref	Reference to UdpNm cluster specific configuration settings. **Tags:** atp.Status=draft
udpPort	PositiveInteger	0..1	attr	This attribute allows to configure a udp port number.

Table C.41: EthernetNetworkConfiguration

Class	**EthernetPhysicalChannel**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	The EthernetPhysicalChannel represents a VLAN or an untagged channel. An untagged channel is modeled as an EthernetPhysicalChannel without an aggregated VLAN. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhysicalChannel*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
networkEndpoint	NetworkEndpoint	*	aggr	Collection of NetworkEndpoints that are used in the VLan. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName
vlan	VlanConfig	0..1	aggr	VLAN Configuration.

Table C.42: EthernetPhysicalChannel

Class	***FibexElement*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore			
Note	ASAM FIBEX elements specifying Communication and Topology.			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*BusMirrorChannelMapping*, *CommunicationCluster*, CouplingElement, EcuInstance, *Frame*, Gateway, GlobalTimeDomain, ISignal, ISignalGroup, ISignalIPduGroup, MachineDesign, NmConfig, *Pdu*, PdurIPduGroup, SecureCommunicationPropsSet, SoAdRoutingGroup, *TpConfig*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.43: FibexElement

Class	**GlobalTimeDomain**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the ability to define a global time domain. **Tags:** atp.recommendedPackage=GlobalTimeDomains			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
debounceTime	TimeValue	0..1	attr	Defines the minimum amount of time between two time sync messages are transmitted.
domainId	PositiveInteger	1	attr	This represents the ID of the GlobalTimeDomain used in the network messages sent on behalf of global time management.
gateway	GlobalTimeGateway	*	aggr	A GlobalTimeGateway may exist in the context of a GlobalTimeDomain to actively update the global time information as it is routed from one GlobalTimeDomain to another. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
globalTimeCorrectionProps	GlobalTimeCorrectionProps	0..1	aggr	Defintion of attributes for rate and offset correction.
globalTimeDomainProperty	AbstractGlobalTimeDomainProps	0..1	aggr	Additional properties of the GlobalTimeDomain. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
globalTimeMaster	GlobalTimeMaster	0..1	aggr	This represents the single master of a GlobalTimeDomain. A GlobalTimeDomain may have no GlobalTimeDomain.master, e.g. when it gets its time from a GPS receiver. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
globalTimeSubDomain	GlobalTimeDomain	*	ref	By this means it is possible to create a hierarchy of sub Domains where one global time domain can declare one or more other global time domains as its subDomains. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
offsetTimeDomain	GlobalTimeDomain	0..1	ref	Reference to a synchronized time domain this offset time domain is based on. The reference source is the offset time domain. The reference target is the synchronized time domain.

▽

△

Class	**GlobalTimeDomain**			
pduTriggering	PduTriggering	0..1	ref	This PduTriggering will be taken to transmit the global time information from a GlobalTimeMaster to a the associated GlobalTimeSlaves. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
slave	GlobalTimeSlave	*	aggr	This represents the collections of slaves of the Global TimeDomain. A GlobalTimeDomain may have no GlobalTime Domain.slaves, e.g. when it propagates its time directly to sub domains. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
syncLoss Timeout	TimeValue	0..1	attr	This attribute describes the timeout for the situation that the time synchronization gets lost in the scope of the time domain.

Table C.44: GlobalTimeDomain

Class	***GlobalTimeMaster*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the generic concept of a global time master.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanMaster, GlobalTimeEthMaster, GlobalTimeFrMaster, UserDefinedGlobalTimeMaster			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
communication Connector	Communication Connector	1	ref	The GlobalTimeMaster is bound to the Communication Connector.
immediate ResumeTime	TimeValue	0..1	attr	Defines the minimum time between an "immediate" message and the next periodic message.
isSystemWide GlobalTime Master	Boolean	1	attr	If set to TRUE, the GlobalTimeMaster is supposed to act as the root of global time information.
syncPeriod	TimeValue	1	attr	This represents the period. Unit: seconds

Table C.45: GlobalTimeMaster

Class	***GlobalTimeSlave*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the generic concept of a global time slave.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanSlave, GlobalTimeEthSlave, GlobalTimeFrSlave, UserDefinedGlobalTimeSlave			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
communication Connector	Communication Connector	1	ref	The GlobalTimeSlave is bound to the Communication Connector.
followUp TimeoutValue	TimeValue	0..1	attr	Rx timeout for the follow-up message.
timeLeapFuture Threshold	TimeValue	0..1	attr	Defines the maximum allowed positive difference between the current Local Time Base value and a newly received Global Time Base value.

▽

△

Class	***GlobalTimeSlave*** (abstract)			
timeLeap HealingCounter	PositiveInteger	0..1	attr	Defines the required number of updates to the Time Base where the time difference to the previous received value has to remain within the bounds of timeLeapFuture Threshold and timeLeapPastThreshold until that Time Base is considered healed.
timeLeapPast Threshold	TimeValue	0..1	attr	Defines the maximum allowed negative difference between the current Local Time Base value and a newly received Global Time Base value.

Table C.46: GlobalTimeSlave

Class	***Grant*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class serves as the abstract base class for defining specific Grants **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ComEventGrant, ComFieldGrant, ComFindServiceGrant, ComMethodGrant, ComOfferServiceGrant			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.47: Grant

Enumeration	**HCConditionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management
Note	Defines whether the Health Channel status shall match or not. **Tags:** atp.Status=draft
Literal	***Description***
equal	**Tags:** atp.EnumerationValue=0
notEqual	**Tags:** atp.EnumerationValue=1

Table C.48: HCConditionEnum

Class	**ISignal**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication
Note	Signal of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal is sent in different SignalIPdus to multiple receivers. To support the RTE "signal fan-out" each SignalIPdu contains ISignals. If the same System Signal is to be mapped into several SignalIPdus there is one ISignal needed for each ISignalToIPduMapping. ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping). In case of the SystemSignalGroup an ISignal must be created for each SystemSignal contained in the SystemSignalGroup. **Tags:** atp.recommendedPackage=ISignals
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*

▽

△

Class	ISignal			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
data Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataTransformation, variation Point.shortLabel vh.latestBindingTime=codeGenerationTime
dataTypePolicy	DataTypePolicyEnum	1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=iSignalProps
iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in a UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "Init Value". If a full DataMapping exist for the SystemSignal this information may be available from a configured Sender ComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.
length	Integer	1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.
network Representation Props	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAllignment" and "byteOrder" shall not be used. ▽

▽

△

Class	**ISignal**			
				△ The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalid Value" and the Data Semantics.
systemSignal	SystemSignal	1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal.
timeout Substitution Value	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal.
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class.

Table C.49: ISignal

Class	**ISignalGroup**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	SignalGroup of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal Group is sent in different SignalIPdus to multiple receivers. An ISignalGroup refers to a set of ISignals that shall always be kept together. A ISignalGroup represents a COM Signal Group. Therefore it is recommended to put the ISignalGroup in the same Package as ISignals (see atp.recommendedPackage) **Tags:** atp.recommendedPackage=ISignalGroup			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
comBased SignalGroup Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignalGroup based on the COMBasedTransformer approach. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=comBasedSignalGroup Transformation, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
iSignal	ISignal	*	ref	Reference to a set of ISignals that shall always be kept together.
systemSignal Group	SystemSignalGroup	1	ref	Reference to the SystemSignalGroup that is defined on VFB level and that is supposed to be transmitted in the ISignalGroup.

▽

△

Class	ISignalGroup			
transformationISignalProps	TransformationISignalProps	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignalGroup specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignalGroups are described in the TransformationTechnology class.

Table C.50: ISignalGroup

Class	ISignalIPdu			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
iPduTimingSpecification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
iSignalToPduMapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the Signal IPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
pduCounter	SignalIPduCounter	0..1	aggr	An included Pdu counter is used to ensure that a sequence of Pdus is maintained. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
pduReplication	SignalIPduReplication	0..1	aggr	Pdu Replication is a form of redundancy where the data content of one ISignalIPdu (source) is transmitted inside a set of replica ISignalIPdus. These ISignalIPdus (copies) have different Pdu IDs, identical PduCounters, identical data content and are transmitted with the same frequency. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unusedBitPattern	Integer	1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table C.51: ISignalIPdu

Class	ISignalTriggering			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	A ISignalTriggering allows an assignment of ISignals to physical channels.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
iSignal	ISignal	0..1	ref	This reference shall be used if an ISignal is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignalTriggering-ISignalGroup reference.
iSignalGroup	ISignalGroup	0..1	ref	This reference shall be used if an ISignalGroup is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignal Triggering-ISignal reference.
iSignalPort	ISignalPort	*	ref	References to the ISignalPort on every ECU of the system which sends and/or receives the ISignal. References for both the sender and the receiver side shall be included when the system is completely defined.

Table C.52: ISignalTriggering

Class	IamModuleInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to define a definition of an IAM instantiation. **Tags:** atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModule Instantiation*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
grant	Grant	*	ref	This reference identifies the applicable Grants for this Iam ModuleInstantiation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=grant atp.Status=draft

Table C.53: IamModuleInstantiation

Class	*Identifiable* (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*
Subclasses	ARPackage, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractServiceInstance*, *AdaptiveModuleInstantiation*, AdaptiveSwcInternalBehavior, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartitionMapping, AsynchronousServerCallResultPoint, *AtpBlueprint*, *Atp Blueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, BswInternalTriggeringPoint, BswModuleDependency, *BuildActionEntity*, BuildActionEnvironment, CanTp Address, CanTpChannel, CanTpNode, Chapter, CheckpointTransition, ClassContentConditional, ClientId Definition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *Comm ConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortStructuralElement*, *CryptoServiceMapping*, Data PrototypeGroup, DataTransformation, DdsRpcServiceDeployment, DependencyOnArtifact, Deterministic ▽

▽

△

Class	*Identifiable* (abstract)			
	△ ClientResourceNeeds, *DiagEventDebounceAlgorithm*, DiagnosticConnectedIndicator, DiagnosticData Element, DiagnosticFunctionInhibitSource, DiagnosticMasterToSlaveEventMapping, *DiagnosticRoutine Subfunction*, DoIpLogicAddress, E2EProfileConfiguration, ECUMapping, *EOCExecutableEntityRef Abstract*, EcuPartition, EcucContainerValue, *EcucDefinitionElement*, EcucDestinationUriDef, Ecuc EnumerationLiteralDef, EcucQuery, EcucValidationCondition, End2EndEventProtectionProps, EndToEnd Protection, EventMapping, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeature MapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeature Restriction, FMFeatureSelection, FieldMapping, FireAndForgetMapping, FlatInstanceDescriptor, Flexray ArTpNode, FlexrayTpConnectionControl, FlexrayTpNode, FlexrayTpPduPool, *FrameTriggering*, General Parameter, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, Hw AttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, IPSecRule, IPv6ExtHeaderFilterList, ISignalToI PduMapping, ISignalTriggering, *IdentCaption*, InterfaceMapping, InternalTriggeringPoint, J1939Shared AddressCluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, Mac MulticastGroup, McDataInstance, MemorySection, MethodMapping, ModeDeclaration, ModeDeclaration Mapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, NvBlockDescriptor, *Packageable Element*, ParameterAccess, PduToFrameMapping, PduTriggering, PerInstanceMemory, PersistencyFile Proxy, PersistencyKeyValuePair, PhmAction, *PhmActionItem*, PhmActionList, PhmArbitration, Phm LogicalExpression, PhmRule, *PhmSupervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, PossibleErrorReaction, ProcessDesignToMachineDesignMapping, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, ResourceConsumption, ResourceGroup, *Rest AbstractEndpoint*, RestElementDef, RestResourceDef, RootSwComponentPrototype, RootSw CompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntityGroup, *SdgAttribute*, SdgClass, Sec OcJobMapping, SecOcJobRequirement, *SecureComProps*, SecureCommunicationAuthenticationProps, *SecureCommunicationDeployment*, SecureCommunicationFreshnessProps, *ServerCallPoint*, *Service EventDeployment*, *ServiceFieldDeployment*, ServiceInstanceToSignalMapping, *ServiceInterfaceElement Mapping*, ServiceInterfaceElementSecureComConfig, ServiceInterfaceMapping, *ServiceMethod Deployment*, *ServiceNeeds*, SignalBasedFieldToISignalTriggeringMapping, SocketAddress, Someip EventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *SpecElementReference*, *StackUsage*, StartupConfig, StructuredReq, SupervisionCheckpoint, SwGenericAxisParamType, SwServiceArg, Swc ServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImplMapping, SystemMapping, TcpOptionFilterList, *TimeBaseResource*, TimingCondition, *TimingConstraint*, *Timing Description*, TimingExtensionResource, TimingModeInstance, TlsCryptoCipherSuite, TlsJobMapping, Topic1, TpAddress, TraceableText, *TracedFailure*, *TransformationProps*, TransformationPropsToService InterfaceElementMapping, TransformationTechnology, Trigger, VariableAccess, VariationPointProxy, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
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
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:** xml.sequenceOffset=-40

▽

△

Class	***Identifiable*** (abstract)			
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30
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 C.54: Identifiable

Class	**ImplementationDataType**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***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. **Tags:** atp.Status=draft

▽

△

Class	ImplementationDataType			
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table C.55: ImplementationDataType

Class	ImplementationDataTypeElement			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. This element either consists of further subElements or it is further defined via its swDataDefProps. There are several use cases within the system of ImplementationDataTypes fur such a local declaration: • It can represent the elements of an array, defining the element type and array size • It can represent an element of a struct, defining its type • It can be the local declaration of a debug element.			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
arraySize	PositiveInteger	0..1	attr	The existence of this attributes (if bigger than 0) defines the size of an array and declares that this Implementation DataTypeElement represents the type of each single array element. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled in case of a variable size array.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ImplementationDataTypeElement as optional. This means that, at runtime, the ImplementationDataType Element may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end. **Tags:** atp.Status=draft

▽

△

Class	ImplementationDataTypeElement			
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Element of an array, struct, or union in case of a nested declaration (i.e. without using "typedefs"). The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElement.

Table C.56: ImplementationDataTypeElement

Class	MacMulticastGroup			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Per EthernetCluster globally defined MacMulticastGroup. One sender can handle many receivers simultaneously if the receivers have all the same macMulticastAddress. The addresses need to be unique for the particular EthernetCluster. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
macMulticastAddress	MacAddressString	1	attr	A multicast MAC address (Media Access Control address) is a identifier for a group of hosts in a network.

Table C.57: MacMulticastGroup

Primitive	NameToken
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This is an identifier as used in xml, e.g. xml-names. Basic difference to Identifier is the fact that it can contain "-". **Tags:** xml.xsd.customType=NMTOKEN-STRING xml.xsd.type=NMTOKEN

Table C.58: NameToken

Class	***NetworkConfiguration*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class defines the abstract attributes for the configuration of a network for a specific CommunicationConnector. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Referrable*			
Subclasses	EthernetNetworkConfiguration			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.59: NetworkConfiguration

Class	*NonOsModuleInstantiation* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module other than the OS module. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DoIpInstantiation, GenericModuleInstantiation, IamModuleInstantiation, LogAndTraceInstantiation, Nm Instantiation, TimeSyncModuleInstantiation, UcmModuleInstantiation			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table C.60: NonOsModuleInstantiation

Class	*PPortComSpec* (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a provided PortPrototype. This class will contain attributes that are valid for all kinds of provide ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	ModeSwitchSenderComSpec, NvProvideComSpec, ParameterProvideComSpec, PersistencyData ProvidedComSpec, *SenderComSpec*, ServerComSpec			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table C.61: PPortComSpec

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

Table C.62: PPortPrototype

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

Table C.63: PRPortPrototype

Class	***PackageableElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
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*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.64: PackageableElement

Class	***Pdu*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Collection of all Pdus that can be routed through a bus interface.			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	GeneralPurposePdu, *IPdu*, NmPdu, UserDefinedPdu			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hasDynamic Length	Boolean	0..1	attr	This attribute defines whether the Pdu has dynamic length (true) or not (false). Please note that the usage of this attribute is restricted by [constr_3448].
length	Integer	0..1	attr	Pdu length in bytes. In case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length. It should be noted that in former AUTOSAR releases (Rel 2.1, Rel 3.0, Rel 3.1, Rel 4.0 Rev. 1) this parameter was defined in bits. The Pdu length of zero bytes is allowed.

Table C.65: Pdu

Class	**PduTriggering**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The PduTriggering describes on which channel the IPdu is transmitted. The Pdu routing by the PduR is only allowed for subclasses of IPdu. Depending on its relation to entities such channels and clusters it can be unambiguously deduced whether a fan-out is handled by the Pdu router or the Bus Interface. If the fan-out is specified between different clusters it shall be handled by the Pdu Router. If the fan-out is specified between different channels of the same cluster it shall be handled by the Bus Interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
iPdu	Pdu	1	ref	Reference to the Pdu for which the PduTriggering is defined. One I-Pdu can be triggered on different channels (PduR fan-out). The Pdu routing by the PduR is only allowed for subclasses of IPdu. Nevertheless is the reference to the Pdu element necessary since the PduTriggering element is also used to specify the sending and receiving connections to Ecu Ports.

▽

△

Class	**PduTriggering**			
iPduPort	IPduPort	*	ref	References to the IPduPort on every ECU of the system which sends and/or receives the I-PDU. References for both the sender and the receiver side shall be included when the system is completely defined.
iSignal Triggering	ISignalTriggering	*	ref	This reference provides the relationship to the ISignal Triggerings that are implemented by the PduTriggering. The reference is optional since no ISignalTriggering can be defined for DCM and Multiplexed Pdus. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
secOcCrypto Mapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu.
triggerIPduSend Condition	TriggerIPduSend Condition	*	aggr	Defines the trigger for the Com_TriggerIPDUSend API call. Only if all defined TriggerIPduSendConditions evaluate to true (AND associated) the Com_Trigger IPDUSend API shall be called.

Table C.66: PduTriggering

Class	**PhmAction**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defined the Actions and ActionLists for the platform health management. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
action	PhmActionItem	*	aggr	Collection of action items. **Tags:** atp.Status=draft
actionList	PhmActionList	*	aggr	Collection of action lists. **Tags:** atp.Status=draft

Table C.67: PhmAction

Class	***PhmSupervision*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines the base class for various supervisions. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AliveSupervision, DeadlineSupervision, GlobalSupervision, LocalSupervision, LogicalSupervision			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table C.68: PhmSupervision

Class	**PhysicalDimension**			
Package	M2::MSR::AsamHdo::Units			
Note	This class represents a physical dimension. If the physical dimension of two units is identical, then a conversion between them is possible. The conversion between units is related to the definition of the physical dimension. Note that the equivalence of the exponents does not per se define the convertibility. For example Energy and Torque share the same exponents (Nm). Please note further the value of an exponent does not necessarily have to be an integer number. It is also possible that the value yields a rational number, e.g. to compute the square root of a given physical quantity. In this case the exponent value would be a rational number where the numerator value is 1 and the denominator value is 2. **Tags:** atp.recommendedPackage=PhysicalDimensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
currentExp	Numerical	0..1	attr	This attribute represents the exponent of the physical dimension "electric current". **Tags:** xml.sequenceOffset=50
lengthExp	Numerical	0..1	attr	The exponent of the physical dimension "length". **Tags:** xml.sequenceOffset=20
luminous IntensityExp	Numerical	0..1	attr	The exponent of the physical dimension "luminous intensity". **Tags:** xml.sequenceOffset=80
massExp	Numerical	0..1	attr	The exponent of the physical dimension "mass". **Tags:** xml.sequenceOffset=30
molarAmount Exp	Numerical	0..1	attr	The exponent of the physical dimension "quantity of substance". **Tags:** xml.sequenceOffset=70
temperatureExp	Numerical	0..1	attr	The exponent of the physical dimension "temperature". **Tags:** xml.sequenceOffset=60
timeExp	Numerical	0..1	attr	The exponent of the physical dimension "time". **Tags:** xml.sequenceOffset=40

Table C.69: PhysicalDimension

Class	***PlatformHealthManagementInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define a PortInterface for the interaction with Platform Health Management. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	PhmHealthChannelInterface, PhmSupervisedEntityInterface			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table C.70: PlatformHealthManagementInterface

Class	PortGroup			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Group of ports which share a common functionality, e.g. need specific network resources. This information shall be available on the VFB level in order to delegate it properly via compositions. When propagated into the ECU extract, this information is used as input for the configuration of Services like the Communication Manager. A PortGroup is defined locally in a component (which can be a composition) and refers to the "outer" ports belonging to the group as well as to the "inner" groups which propagate this group into the components which are part of a composition. A PortGroup within an atomic SWC cannot be linked to inner groups.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
innerGroup	PortGroup	*	iref	Links a PortGroup in a composition to another PortGroup, that is defined in a component which is part of this CompositionSwComponentType.
outerPort	PortPrototype	*	ref	Outer PortPrototype of this AtomicSwComponentType which belongs to the group. A port can belong to several groups or to no group at all. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table C.71: PortGroup

Class	*PortPrototype* (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
clientServerAnnotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPortAnnotation	DelegatedPortAnnotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstractionServerAnnotation	IoHwAbstractionServerAnnotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePortAnnotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPortAnnotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPortAnnotation	ParameterPortAnnotation	*	aggr	Annotations on this parameter port.
portPrototypeProps	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype. **Tags:** atp.Status=draft
senderReceiverAnnotation	SenderReceiverAnnotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPortAnnotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table C.72: PortPrototype

Class	**ProvidedServiceInstance**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Service instances that are provided by the ECU that is connected via the ApplicationEndpoint to a CommunicationConnector.			
Base	*ARObject*, *AbstractServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
EventHandler	EventHandler	*	aggr	Collection of event callback configurations.
instance Identifier	PositiveInteger	0..1	attr	Instance identifier. Can be used for e.g. service discovery to identify the instance of the service.
loadBalancing Priority	PositiveInteger	0..1	attr	Defines the value to be used for load balancing priority in the service offer. Lower value means higher priority.
loadBalancing Weight	PositiveInteger	0..1	attr	Defines the value to be used for load balancing weight in the service offer. Higher value means higher probability to be chosen.
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.
sdServerConfig	SdServerConfig	0..1	aggr	Service Discovery Server configuration.
serviceIdentifier	PositiveInteger	0..1	attr	Service ID. Shall be unique within one system to allow service discovery.

Table C.73: ProvidedServiceInstance

Class	**PskIdentityToKeySlotMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This meta-class allows to map a PresharedKeyIdentity to a concrete key that will be used for a crypto operation. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.74: PskIdentityToKeySlotMapping

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

Table C.75: RPortComSpec

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

Table C.76: RPortPrototype

Class	RecordValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for a record.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
field (ordered)	ValueSpecification	1..*	aggr	The value for a single record field. This could also be mapped explicitly to a record element of the data type using the shortName of the ValueSpecification. But this would introduce a relationship to the data type that is too strong. As of now, it is only important that the structure of the data type matches the structure of the Value Specification indepenently of the shortNames. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table C.77: RecordValueSpecification

Primitive	Ref			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	This primitive denotes a name based reference. For detailed syntax see the xsd.pattern. • first slash (relative or absolute reference) [optional] • Identifier [required] • a sequence of slashes and Identifiers [optional] This primitive is used by the meta-model tools to create the references. **Tags:** xml.xsd.customType=REF xml.xsd.pattern=/?[a-zA-Z][a-zA-Z0-9_]{0,127}(/[a-zA-Z][a-zA-Z0-9_]{0,127})* xml.xsd.type=string			
Attribute	**Datatype**	**Mul.**	**Kind**	**Note**
base	Identifier	0..1	attr	This attribute reflects the base to be used for this reference. **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:** atp.Status=draft xml.attribute=true

▽

△

Primitive	**Ref**			
index	PositiveInteger	0..1	attr	This attribute supports the use case to point on specific elements in an array. This is in particular required if arrays are used to implement particular data objects. **Tags:** xml.attribute=true

Table C.78: Ref

Class	**ReferenceValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies a reference to a data prototype to be used as an initial value for a pointer in the software.			
Base	*ARObject*, *ValueSpecification*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
referenceValue	DataPrototype	1	ref	The referenced data prototype.

Table C.79: ReferenceValueSpecification

Class	***Referrable*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, Bsw VariableAccess, CouplingPortTrafficClassAssignment, *CppImplementationDataTypeContextTarget*, DiagnosticDebounceAlgorithmProps, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, Event Handler, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, *NetworkConfiguration*, NmNetworkHandle, PncMappingIdent, *SingleLanguageReferrable*, SocketConnectionBundle, SomeipRequiredEventGroup, TimeSyncServer Configuration, TpConnectionIdent			
Attribute	***Type***	***Mul.***	***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. **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 C.80: Referrable

Primitive	**RevisionLabelString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This primitive represents a revision label which identifies an engineering object. It represents a pattern which • requires three integers representing from left to right MajorVersion, MinorVersion, PatchVersion. • may add an application specific suffix separated by one of ".", "_", ";". Legal patterns are for example: ▽

▽

△

Primitive	**RevisionLabelString**
	△ 4.0.0 4.0.0.1234565 4.0.0_vendor specific;13 4.0.0;12 **Tags:** xml.xsd.customType=REVISION-LABEL-STRING xml.xsd.pattern=[0-9]+\.[0-9]+\.[0-9]+([\._;].*)? xml.xsd.type=string

Table C.81: RevisionLabelString

Class	**RoleBasedPortAssignment**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
Note	This class specifies an assignment of a role to a particular service port (RPortPrototype or PPort Prototype) of an AtomicSwComponentType. With this assignment, the role of the service port can be mapped to a specific ServiceNeeds element, so that a tool is able to create the correct connector.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
portPrototype	PortPrototype	1	ref	Service PortPrototype used in the assigned role. This PortPrototype shall either belong to the same AtomicSw ComponentType as the SwcInternalBehavior which owns the ServiceDependency or to the same NvBlockSw ComponentType as the NvBlockDescriptor.
role	Identifier	1	attr	This is the role of the assigned Port in the given context. The value shall be a shortName of the Blueprint of a Port Interface as standardized in the Software Specification of the related AUTOSAR Service.

Table C.82: RoleBasedPortAssignment

Class	**Sd**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This class represents a primitive element in a special data group.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:** xml.attribute=true
value	VerbatimStringPlain	1	attr	This is the value of the special data. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.typeElement=false xml.typeWrapperElement=false

▽

△

Class	**Sd**			
xmlSpace	XmlSpaceEnum	0..1	attr	This attribute is used to signal an intention that in that element, white space should be preserved by applications. It is defined according to xml:space as declared by W3C. **Tags:** xml.attribute=true xml.attributeRef=true xml.enforceMinMultiplicity=true xml.name=space xml.nsPrefix=xml

Table C.83: Sd

Class	**Sdg**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	Sdg (SpecialDataGroup) is a generic model which can be used to keep arbitrary information which is not explicitly modeled in the meta-model. Sdg can have various contents as defined by sdgContentsType. Special Data should only be used moderately since all elements should be defined in the meta-model. Thereby SDG should be considered as a temporary solution when no explicit model is available. If an sdg Caption is available, it is possible to establish a reference to the sdg structure.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:** xml.attribute=true
sdgCaption	SdgCaption	0..1	aggr	This aggregation allows to assign the properties of Identifiable to the sdg. By this, a shortName etc. can be assigned to the Sdg. **Tags:** xml.sequenceOffset=20
sdgCaptionRef	SdgCaption	0..1	ref	This association allows to reuse an already existing caption. **Tags:** xml.name=SDG-CAPTION-REF xml.sequenceOffset=25
sdgContents Type	SdgContents	0..1	aggr	This is the content of the Sdg. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table C.84: Sdg

Class	**SecOcDeployment**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	The meta-class represents the ability to define a deployment of the SecOc communication protocol configuration settings to crypto module entities. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft

▽

△

Class	SecOcDeployment			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SecureCommunicationDeployment*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
secOcJob Mapping	SecOcJobMapping	*	aggr	Mapping of the JobRequirement to a concrete crypto job. **Tags:** atp.Status=draft

Table C.85: SecOcDeployment

Class	SecOcJobMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This meta-class allows to map a SecOcJobRequirement to a concrete crypto job that will fulfill the Job Requirement. The crypto job represents a call to a specific routine that implements a crypto function and that uses a specific key and refers to a specific primitive as a formal representation of the crypto algorithm. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
secOcJob Requirement	SecOcJobRequirement	1	ref	Reference to a SecOC JobRequirement that defines requirements for the cryptographic job that need to be executed. **Tags:** atp.Status=draft

Table C.86: SecOcJobMapping

Class	***SecureCommunicationDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	The meta-class represents the ability to define a deployment of secure communication protocol configuration settings to crypto module entities. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SecOcDeployment, TlsDeployment			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table C.87: SecureCommunicationDeployment

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

▽

△

Class	SenderReceiverInterface			
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement

Table C.88: SenderReceiverInterface

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

Table C.89: ServerComSpec

Class	***ServiceNeeds*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This expresses the abstract needs that a Software Component or Basic Software Module has on the configuration of an AUTOSAR Service to which it will be connected. "Abstract needs" means that the model abstracts from the Configuration Parameters of the underlying Basic Software.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswMgrNeeds, ComMgrUserNeeds, CryptoServiceJobNeeds, CryptoServiceNeeds, *Diagnostic CapabilityElement*, DltUserNeeds, *DoIpServiceNeeds*, EcuStateMgrUserNeeds, ErrorTracerNeeds, FunctionInhibitionAvailabilityNeeds, FunctionInhibitionNeeds, GlobalSupervisionNeeds, HardwareTest Needs, IndicatorStatusNeeds, J1939RmIncomingRequestServiceNeeds, J1939RmOutgoingRequest ServiceNeeds, NvBlockNeeds, SecureOnBoardCommunicationNeeds, SupervisedEntityCheckpoint Needs, SupervisedEntityNeeds, SyncTimeBaseMgrUserNeeds, V2xFacUserNeeds, V2xMUserNeeds, VendorSpecificServiceNeeds			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.90: ServiceNeeds

Class	**ServiceSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	ServiceSwComponentType is used for configuring services for a given ECU. Instances of this class are only to be created in ECU Configuration phase for the specific purpose of the service configuration. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *Atp Type*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *Sw ComponentType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.91: ServiceSwComponentType

Class	**SocketConnection**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetCommunication			
Note	The SoAd serves as a (De)Multiplexer between different PDU sources and the TCP/IP stack.			
Base	*ARObject*, *Describable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
allowedIPv6ExtHeaders	IPv6ExtHeaderFilterList	0..1	ref	Reference to a list of IPv6 Extension Headers allowed for this SocketConnection. If no list is referenced all IPv6 Extension Headers are allowed and processed.
allowedTcpOptions	TcpOptionFilterList	0..1	ref	Reference to a list of TCP options allowed for this SocketConnection.
clientIpAddrFromConnectionRequest	Boolean	0..1	attr	If set to true the Server "learns" the client IP address on connection request. This means that the statically configured IP Address of the related client shall be ignored. If set to false the Server only accepts statically configured IP address, e.g. 192.168.1.2. This means that the statically configured IP Address of the Client shall be used.
clientPort	SocketAddress	0..1	ref	Client Port for TCP/UDP connection in an abstract communication sense. The client is the major requester of the communication. Please note that the client may also produce data.
clientPortFromConnectionRequest	Boolean	0..1	attr	If set to true the Server "learns" the client Port on connection request. This means that the statically configured Port of the related client shall be ignored. If set to false the Server only accepts statically configured Port. This means that the statically configured Port of the Client shall be used.
pdu	SocketConnectionIpduIdentifier	*	aggr	PDUs handed over by the PDU Router (Transmission over the Ethernet) or PDUs handed over by SoAd (Reception over Ethernet). Multiple IPdus can be transmitted over one socket connection.
pduCollectionMaxBufferSize	PositiveInteger	0..1	attr	Defines the maximum buffer size in Byte which shall be filled before a socket with Pdu collection enabled shall be transmitted to the lower layer.
pduCollectionTimeout	TimeValue	0..1	attr	Defines the time in seconds which shall pass before a socket with Pdu collection enabled shall be transmitted to the lower layer after the first Pdu has been put into the socket buffer.
runtimeIpAddressConfiguration	RuntimeAddressConfigurationEnum	0..1	attr	This attribute determines which protocol is used by the client to obtain the IP Address information. If this attribute is not set to none the value determines the service used by the client to obtain the IP Address information for the SocketConnection. If this attribute is set to none the client used the statically configured IP Address information.
runtimePortConfiguration	RuntimeAddressConfigurationEnum	0..1	attr	This attribute determines which protocol is used by the client to obtain the Port information. If this attribute is not set to none the value determines the service used by the client to obtain the Port information for the SocketConnection. If this attribute is set to none the client uses the statically configured Port information.
shortLabel	Identifier	0..1	attr	This attribute specifies an identifying shortName for the SocketConnection. It shall be unique within its context.

Table C.92: SocketConnection

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

Table C.93: SwBaseType

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

Table C.94: SwComponentPrototype

Class	***SwConnector*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	The base class for connectors between ports. Connectors have to be identifiable to allow references from the system constraint template.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AssemblySwConnector, DelegationSwConnector, PassThroughSwConnector			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
mapping	PortInterfaceMapping	0..1	ref	Reference to a PortInterfaceMapping specifying the mapping of unequal named PortInterface elements of the two different PortInterfaces typing the two PortPrototypes which are referenced by the ConnectorPrototype.

Table C.95: SwConnector

Class	«atpVariation» **SwDataDefProps**
Package	M2::MSR::DataDictionary::DataDefProperties
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the Data ▽

▽

△

Class	«atpVariation» **SwDataDefProps**			
	△ Types in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddrMethod, swPointerTagetProps, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue • Code generation policy provided by swRecordLayout **Tags:** vh.latestBindingTime=codeGenerationTime			
Base	*ARObject*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
additionalNativeTypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. **Tags:** xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:** xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:** xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:** xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:** xml.sequenceOffset=210
displayPresentation	DisplayPresentationEnum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementationDataType	AbstractImplementationDataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTargetProps), if it does not refer to a base type directly ▽

▽

△

Class	«atpVariation» **SwDataDefProps**			
				△ • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:** xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:** xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:** xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memory AllocationKeywordPolicy of the referenced SwAddr Method. **Tags:** xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:** xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:** xml.sequenceOffset=70
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:** xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170 xml.typeElement=false
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:** xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:** xml.sequenceOffset=230

▽

△

Class	«atpVariation» **SwDataDefProps**			
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:** xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:** xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:** xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:** xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:** xml.sequenceOffset=290
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:** xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:** xml.sequenceOffset=120
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock Size Mult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension. The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on. ▽

▽

△

Class	«atpVariation» **SwDataDefProps**			
				△ For one-dimensional value blocks the attribute swValueBlockSize shall be used and this attribute shall not exist. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:** xml.sequenceOffset=350
valueAxisDataType	ApplicationPrimitiveDataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:** xml.sequenceOffset=355

Table C.96: SwDataDefProps

Class	**SwPointerTargetProps**			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This element defines, that the data object (which is specified by the aggregating element) contains a reference to another data object or to a function in the CPU code. This corresponds to a pointer in the C-language. The attributes of this element describe the category and the detailed properties of the target which is either a data description or a function signature.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
functionPointerSignature	BswModuleEntry	0..1	ref	The referenced BswModuleEntry serves as the signature of a function pointer definition. Primary use case: function pointer passed as argument to other function. **Tags:** xml.sequenceOffset=40
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of the target data type. **Tags:** xml.sequenceOffset=30
targetCategory	Identifier	0..1	attr	This specifies the category of the target: • In case of a data pointer, it shall specify the category of the referenced data. • In case of a function pointer, it could be used to denote the category of the referenced BswModuleEntry. Since currently no categories for BswModuleEntry are defined it will be empty. **Tags:** xml.sequenceOffset=5

Table C.97: SwPointerTargetProps

Class	SwRecordLayout			
Package	M2::MSR::DataDictionary::RecordLayout			
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. **Tags:** atp.recommendedPackage=SwRecordLayouts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	Type	Mul.	Kind	Note
swRecord LayoutGroup	SwRecordLayoutGroup	1	aggr	This is the top level record layout group. **Tags:** xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table C.98: SwRecordLayout

Class	SystemSignal			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The system signal represents the communication system's view of data exchanged between SW components which reside on different ECUs. The system signals allow to represent this communication in a flattened structure, with exactly one system signal defined for each data element prototype sent and received by connected SW component instances. **Tags:** atp.recommendedPackage=SystemSignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	Type	Mul.	Kind	Note
dynamicLength	Boolean	1	attr	The length of dynamic length signals is variable in run-time. Only a maximum length of such a signal is specified in the configuration (attribute length in ISignal element).
physicalProps	SwDataDefProps	0..1	aggr	Specification of the physical representation.

Table C.99: SystemSignal

Class	*TimeSynchronizationInterface* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define a PortInterface for the interaction with Time Synchronization. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	TimeSynchronizationMasterInterface, TimeSynchronizationPureLocalInterface, TimeSynchronization SlaveInterface			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table C.100: TimeSynchronizationInterface

Class	**TlsCryptoServiceMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto service mapping for the socket-based configuration of Transport Layer Security (TLS). **Tags:** atp.Status=draft atp.recommendedPackage=CryptoServiceMappings			
Base	*ARObject*, *CryptoServiceMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared(i.e. applicable for each of the aggreated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase. **Tags:** atp.Status=draft
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	This aggregation represents the collection of supported cipher suites. **Tags:** atp.Status=draft

Table C.101: TlsCryptoServiceMapping

Class	**TlsDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	The meta-class represents the ability to define a deployment of the TLS communication protocol configuration settings to crypto module entities. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SecureCommunicationDeployment*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
pskIdentityToKeySlotMapping	PskIdentityToKeySlotMapping	*	aggr	Mapping of TLS-PSK to a concrety key defined in the CryptoDeployment. **Tags:** atp.Status=draft
tlsJobMapping	TlsJobMapping	*	aggr	Mapping of the JobRequirement to a concrete crypto job. **Tags:** atp.Status=draft

Table C.102: TlsDeployment

Class	**TlsJobMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This meta-class allows to map a TlsJobRequirement to a concrete crypto job that will fulfill the Job Requirement. The crypto job represents a call to a specific routine that implements a crypto function and that uses a specific key and refers to a specific primitive as a formal representation of the crypto algorithm. **Tags:** atp.ManifestKind=MachineManifest atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.103: TlsJobMapping

Class	*TransformationProps* (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class represents a abstract base class for transformation settings.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ApSomeipTransformationProps, SOMEIPTransformationProps, UserDefinedTransformationProps			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table C.104: TransformationProps

Enumeration	TransportLayerProtocolEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	This enumeration allows to choose a TCP/IP transport layer protocol. **Tags:** atp.Status=draft
Literal	**Description**
tcp	Transmission control protocol **Tags:** atp.EnumerationValue=1
udp	User datagram protocol **Tags:** atp.EnumerationValue=0

Table C.105: TransportLayerProtocolEnum

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

Table C.106: Trigger

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

Table C.107: TriggerInterface

Class	**Unit**			
Package	M2::MSR::AsamHdo::Units			

Note	This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined.
	For the calculation from SI-unit to the defined unit the factor (factorSiToUnit) and the offset (offsetSiToUnit) are applied as follows:
	`x [{unit}] := y * [{siUnit}] * factorSiToUnit [[unit]/{siUnit}] + of`
	For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit) and the negation of the offset (offsetSiToUnit) are applied.
	`y {siUnit} := (x*{unit} - offsetSiToUnit [{unit}]) / (factorSiToUnit [[u`
	Tags: atp.recommendedPackage=Units			

Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
displayName	SingleLanguageUnit Names	0..1	aggr	This specifies how the unit shall be displayed in documents or in user interfaces of tools.The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file. **Tags:** xml.sequenceOffset=20
factorSiToUnit	Float	0..1	attr	This is the factor for the conversion from SI Units to units. The inverse is used for conversion from units to SI Units. **Tags:** xml.sequenceOffset=30
offsetSiToUnit	Float	0..1	attr	This is the offset for the conversion from and to siUnits. **Tags:** xml.sequenceOffset=40
physical Dimension	PhysicalDimension	0..1	ref	This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted. **Tags:** xml.sequenceOffset=50

Table C.108: Unit

Class	**UserDefinedServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map UserDefinedServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.ManifestKind=ServiceInstanceManifest atp.Status=draft atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.109: UserDefinedServiceInstanceToMachineMapping

Class	*ValueSpecification* (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Base class for expressions leading to a value which can be used to initialize a data object.			
Base	*ARObject*			
Subclasses	*AbstractRuleBasedValueSpecification*, ApplicationValueSpecification, *CompositeValueSpecification*, ConstantReference, NotAvailableValueSpecification, NumericalValueSpecification, ReferenceValueSpecification, TextValueSpecification			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
shortLabel	Identifier	0..1	attr	This can be used to identify particular value specifications for human readers, for example elements of a record type.

Table C.110: ValueSpecification

D History of Constraints and Specification Items

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

D.1 Constraint History of this Document according to the original version of the Document

D.1.1 Created Constraints

Number	Heading
[constr_1473]	No support for `PRPortPrototype`
[constr_1474]	`SwDataDefProps` applicable to `ImplementationDataType`s exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	`ImplementationDataType` of `category` `STRING` is limited
[constr_1476]	`ImplementationDataType` of `category` `VECTOR` is limited
[constr_1477]	`ImplementationDataType` of `category` `ASSOCIATIVE_MAP` is limited
[constr_1478]	`SwDataDefProps` applicable to `ApplicationDataType`s exclusive to the *AUTOSAR adaptive platform*
[constr_1479]	No support for certain values of `ImplementationDataType.category`
[constr_1480]	Mutual existence of CompositionDataPrototypeRef.elementInImplDatatype vs. attributes of CompositionDataPrototypeRef.dataPrototype
[constr_1481]	Usage of CompositionDataPrototypeRef in the *AUTOSAR adaptive platform*
[constr_1482]	Mapping of service interfaces vs. mapping of service interface elements
[constr_1483]	Applicability of a `ServiceInterface`

▽

△

Number	Heading
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1485]	No subElement for ImplementationDataType of category STRING
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1487]	Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP
[constr_1488]	Initialization of a DataPrototype typed by an ApplicationAssocMapDataType
[constr_1489]	Uniqueness of ApplicationAssocMapValueSpecification.mapElementTuple.key
[constr_1490]	Allowed value of category for reference AdaptiveModuleInstantiation.process.executable
[constr_1491]	Reference to ApplicationError
[constr_1492]	SwComponentType referenced as Executable.rootSwComponentPrototype.applicationType
[constr_1493]	ArgumentDataPrototype referenced in the role ApplicationError.errorContext
[constr_1494]	Initial value for event
[constr_1495]	Initial value for field
[constr_1496]	DiagnosticServiceDataMapping.mappedApDataElement shall only refer to specific sub-classes of DataPrototype
[constr_1497]	Attribute optionKind set to commandLineSimpleForm
[constr_1498]	Attribute optionKind set to commandLineShortForm or commandLineLongForm
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable
[constr_1501]	Target SwcServiceDependency of DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable
[constr_1502]	Target SwcServiceDependency of DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable
[constr_1503]	Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same ModeDeclaration
[constr_1505]	Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3287]	Mandatory information of a ProvidedSomeipServiceInstance
[constr_3288]	IP configuration restriction for unicastNetworkEndpoints

▽

△

Number	Heading
[constr_3290]	Usage of ServiceInstancePortConfig defined for a ProvidedSomeipServiceInstance
[constr_3291]	SomeipServiceInstanceToMachineMapping.portConfig aggregation restriction
[constr_3293]	Mandatory information of a RequiredSomeipServiceInstance
[constr_3296]	Usage of ServiceInstancePortConfig defined for a RequiredSomeipServiceInstance
[constr_3297]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_3300]	Allowed ServiceMethodDeployment.method references
[constr_3301]	Allowed ServiceEventDeployment.event references
[constr_3302]	Allowed ServiceFieldDeployment.field references
[constr_3303]	ANY not allowed for SomeipServiceInterface.serviceInterfaceVersion
[constr_3304]	Value of attribute SomeipEventGroup.eventGroupId shall be unique
[constr_3305]	Value of attribute SomeipEvent.eventId shall be unique
[constr_3306]	Value of attribute SomeipMethod.methodId shall be unique
[constr_3307]	SomeipEvent.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEvent.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethod.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethod.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3349]	Usage of ApplicationAssocMapDataType is limited
[constr_3350]	Consistent value of category for AdaptiveAutosarApplications referencing an Executable
[constr_3351]	SOME/IP segmentation allowed for udp SomeipEvents
[constr_3352]	SOME/IP segmentation allowed for udp SomeipMethods
[constr_3353]	Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLengthField
[constr_3354]	Restriction in usage of ApSomeipTransformationProps.sizeOfStructLengthField
[constr_3355]	Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLengthField
[constr_3356]	Restriction in usage of ApSomeipTransformationProps.alignment
[constr_3357]	Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField
[constr_3358]	Usage of PortPrototype and TransportLayerIndependentInstanceId to define the same Service Instance is not allowed.
[constr_3359]	RPortPrototypeProps are related only to RPortPrototypes.

▽

△

Number	Heading
[constr_3360]	RPortPrototypeProps are related only to TransportLayerIndependentInstanceIds representing a consumer Service Instance.
[constr_3361]	Selective definition of serialization settings.
[constr_3362]	SomeipEvents aggregated by a SomeipField
[constr_3363]	SomeipMethods aggregated by a SomeipField

Table D.1: Added Constraints in original version

D.1.2 Created Specification Items

Number	Heading
[TPS_MANI_01000]	Definition of the term Manifest
[TPS_MANI_01001]	Meaning of ServiceInterface
[TPS_MANI_01002]	Semantics of a ServiceInterfaceMapping
[TPS_MANI_01003]	Limitations of the applicability of ServiceInterfaceMapping
[TPS_MANI_01004]	Semantics of ServiceInterface.namespace
[TPS_MANI_01005]	The definition of the namespace of a ServiceInterface may follow a hierarchical pattern
[TPS_MANI_01006]	Ordered definition of ServiceInterface.namespace
[TPS_MANI_01007]	Service-oriented **communication** and service **discovery**
[TPS_MANI_01008]	Semantics of AdaptiveAutosarApplication
[TPS_MANI_01009]	Standardized values of AdaptiveAutosarApplication.category
[TPS_MANI_01010]	Root element for a hierarchical software-component
[TPS_MANI_01011]	Connection between application design and application deployment
[TPS_MANI_01012]	Formal modeling of application startup behavior
[TPS_MANI_01013]	Semantics of meta-class ModeDependentStartupConfig
[TPS_MANI_01014]	Semantics of meta-class StartupConfigSet
[TPS_MANI_01015]	Semantics of meta-class StartupOption
[TPS_MANI_01016]	Category of ApplicationAssocMapDataType
[TPS_MANI_01017]	Relation of startup configuration to resource groups
[TPS_MANI_01018]	ImplementationDataType of category VECTOR
[TPS_MANI_01019]	Manifest content may apply to different aspects of the *AUTOSAR adaptive platform*
[TPS_MANI_01020]	Serialization format of the *Manifest* in AUTOSAR
[TPS_MANI_01021]	Serialization format of *Manifest* content on a machine
[TPS_MANI_01022]	Concept behind ServiceInterfaceMapping
[TPS_MANI_01024]	Semantics of ServiceInterfaceEventMapping
[TPS_MANI_01025]	Semantics of ServiceInterfaceFieldMapping

▽

△

Number	Heading
[TPS_MANI_01026]	Semantics of ServiceInterfaceMethodMapping
[TPS_MANI_01027]	Semantics of ApplicationAssocMapDataType
[TPS_MANI_01028]	ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01029]	Usage of ImplementationDataType
[TPS_MANI_01030]	ImplementationDataType of category STRING
[TPS_MANI_01031]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01032]	Usage of ServiceInterfaceMapping
[TPS_MANI_01033]	Semantics of ServiceInterface.event
[TPS_MANI_01034]	Semantics of ServiceInterface.field
[TPS_MANI_01035]	Semantics of ServiceInterface.method
[TPS_MANI_01037]	Diagnostic data mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01038]	Diagnostic software mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01039]	Representation of provided service
[TPS_MANI_01040]	Representation of required service
[TPS_MANI_01041]	Startup configuration supports the definition of a launch dependency
[TPS_MANI_01042]	Definition of a linear ImplementationDataType of category VECTOR
[TPS_MANI_01043]	Definition of a rectangular ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01045]	Process.modeDependentStartupConfig that does not refer to a ModeDeclaration
[TPS_MANI_01046]	Semantics of ModeDependentStartupConfig.machineMode
[TPS_MANI_01047]	Existence of SwRecordLayout for an ApplicationPrimitiveDataType of category STRING
[TPS_MANI_01048]	Mapping of DiagnosticEvent to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01049]	Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of DiagnosticEnableCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01052]	Semantics of RPortPrototypeProps.portInstantiationBehavior
[TPS_MANI_01053]	Usage of ComSpecs on the *AUTOSAR adaptive platform*
[TPS_MANI_01054]	Definition of the queue length of an event
[TPS_MANI_01055]	Semantics of ServiceInterface.possibleError
[TPS_MANI_01056]	Semantics of ApplicationError.errorContext
[TPS_MANI_01057]	Semantics of RPortPrototypeProps.searchBehavior

▽

△

Number	Heading
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01059]	Different values of optionKind within a StartupConfig.startupOption
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_03000]	Mapping of AdaptivePlatformServiceInstance to PortPrototypes
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a Machine
[TPS_MANI_03002]	IP configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03003]	ProvidedSomeipServiceInstance Fanout
[TPS_MANI_03004]	IPv4 Multicast event destination address
[TPS_MANI_03005]	IPv4 Multicast address range
[TPS_MANI_03006]	IPv6 Multicast address range
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03009]	Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03011]	Server Timing configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03012]	Initial Wait Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03013]	Repetition Wait Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03014]	Main Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03015]	TTL for Offer Service Entries
[TPS_MANI_03016]	Servers RequestResponseDelay for received FindService entries
[TPS_MANI_03017]	Server Capability Records
[TPS_MANI_03018]	Usage of SomeipProvidedEventGroup.multicastThreshold
[TPS_MANI_03019]	TTL for SubscribeEventGroupAck Entries
[TPS_MANI_03020]	Servers RequestResponseDelay for received SubscribeEventGroup entries
[TPS_MANI_03021]	Requirements on the service version from the client's point of view
[TPS_MANI_03022]	Context of RequiredSomeipServiceInstance
[TPS_MANI_03023]	Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03024]	Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03025]	Client Timing configuration for a RequiredSomeipServiceInstance

▽

△

Number	Heading
[TPS_MANI_03026]	Initial Wait Phase configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03027]	Repetition Wait Phase configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03028]	TTL for Find Service Entries
[TPS_MANI_03029]	Client Capability Records
[TPS_MANI_03030]	SomeipSdClientEventGroupTimingConfig.timeToLive for SubscribeEventGroup Entries
[TPS_MANI_03031]	Clients RequestResponseDelay for received ServiceOffer entries
[TPS_MANI_03032]	Description of middleware technologies not standardized by AUTOSAR
[TPS_MANI_03035]	Content of the Machine configuration
[TPS_MANI_03036]	ServiceInterface deployment to a middleware transport layer
[TPS_MANI_03037]	Purpose of ServiceMethodDeployment
[TPS_MANI_03038]	Purpose of ServiceEventDeployment
[TPS_MANI_03039]	Purpose of ServiceFieldDeployment
[TPS_MANI_03040]	SOME/IP ServiceInterface binding
[TPS_MANI_03041]	Definition of SOME/IP EventGroups
[TPS_MANI_03042]	Definition of SOME/IP Service Version
[TPS_MANI_03043]	SOME/IP VariableDataPrototype binding
[TPS_MANI_03044]	SOME/IP ClientServerOperation binding
[TPS_MANI_03045]	UserDefined ServiceInterface binding
[TPS_MANI_03046]	User defined VariableDataPrototype binding
[TPS_MANI_03047]	User defined ClientServerOperation binding
[TPS_MANI_03048]	User defined Field binding
[TPS_MANI_03049]	Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03050]	Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03051]	Usage of SomeipMethod.transportProtocol
[TPS_MANI_03052]	Static IPv4 configuration
[TPS_MANI_03053]	Static IPv6 configuration
[TPS_MANI_03056]	Usage of SomeipEvent.transportProtocol
[TPS_MANI_03057]	SOME/IP Field binding
[TPS_MANI_03059]	RequiredSomeipServiceInstance.requiredServiceInstanceId
[TPS_MANI_03061]	IPv6 Multicast event destination address
[TPS_MANI_03064]	SOME/IP Service Discovery message exchange configuration
[TPS_MANI_03065]	Hardware resources of the machine
[TPS_MANI_03066]	Description of machine states
[TPS_MANI_03067]	SOME/IP segmentation of udp SomeipEvents

▽

△

Number	Heading
[TPS_MANI_03068]	SOME/IP segmentation of SomeipMethod Calls
[TPS_MANI_03069]	SOME/IP segmentation of SomeipMethod Responses
[TPS_MANI_03070]	Size of a length field for a chosen array
[TPS_MANI_03071]	Size of a length field for a chosen structure
[TPS_MANI_03072]	Size of a length field for a chosen union
[TPS_MANI_03073]	Alignment of a dynamic DataPrototype
[TPS_MANI_03074]	Size of a type selector field for a chosen union
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03094]	Machine-specific platform configuration settings
[TPS_MANI_03095]	Implementation-specific platform configuration settings
[TPS_MANI_03096]	Machine-specific configuration settings for a generic module
[TPS_MANI_03097]	Implementation-specific configuration settings for a generic module
[TPS_MANI_03098]	Machine-specific configuration settings for the OS module
[TPS_MANI_03099]	Implementation-specific configuration settings for the OS module
[TPS_MANI_03100]	Transport layer independent TransportLayerIndependentInstanceIds
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array length fields
[TPS_MANI_03104]	Default size for all structure length fields
[TPS_MANI_03105]	Default size for all union length fields
[TPS_MANI_03106]	Default size for all union type selector fields
[TPS_MANI_03107]	Default alignment for all dynamic DataPrototypes
[TPS_MANI_03108]	Default Byte Order for all DataPrototypes
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface

Table D.2: Added Specification Items in original Version

D.2 Constraint and Specification Item History of this document according to AUTOSAR Release 17-10

D.2.1 Added Traceables in 17-10

Number	Heading
[TPS_MANI_01064]	Semantics of attribute method.fireAndForget
[TPS_MANI_01065]	Purpose of PersistencyKeyValueDatabaseInterface
[TPS_MANI_01067]	Purpose of PersistencyFileProxyInterface

▽

△

Number	Heading
[TPS_MANI_01068]	Semantics of PersistencyFileProxyInterface.maxNumberOfFiles
[TPS_MANI_01069]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces
[TPS_MANI_01073]	Semantics of PortPrototype typed by PersistencyKeyValueDatabaseInterface
[TPS_MANI_01074]	Specification of encryption of persistent data
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01077]	Specification of file encryption
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping
[TPS_MANI_01079]	Semantics of PersistencyKeyValueDatabase
[TPS_MANI_01080]	Semantics of PersistencyFileProxyToFileMapping
[TPS_MANI_01081]	Semantics of PortPrototype typed by PersistencyFileProxyInterface
[TPS_MANI_01082]	Eligibility of DataPrototypes for the definition of optionality
[TPS_MANI_01083]	Optionality is supported for ApplicationDataType as well as ImplementationDataType
[TPS_MANI_01084]	Optionality for a DataPrototype typed by an ApplicationDataType
[TPS_MANI_01085]	Definition of optionality for a DataPrototype typed by an ImplementationDataType
[TPS_MANI_01087]	Interaction with crypto software
[TPS_MANI_01088]	Semantics of CryptoNeed
[TPS_MANI_01089]	Relation between CryptoNeed and PortPrototype
[TPS_MANI_01090]	Modeling of crypto software as a platform module
[TPS_MANI_01091]	Semantics of CryptoJob
[TPS_MANI_01092]	Mapping between CryptoNeed and CryptoJob
[TPS_MANI_01093]	Semantics of CryptoDriver
[TPS_MANI_01094]	Scope of CryptoDriver
[TPS_MANI_01095]	Semantics of CryptoKeySlot
[TPS_MANI_01096]	Semantics of the CryptoPrimitive
[TPS_MANI_01097]	Assignment of TLV data ids for data structures with optional members
[TPS_MANI_01098]	Constraints on the definition of an ImplementationDataType of category VECTOR
[TPS_MANI_01099]	Semantics of ImplementationDataTypeElementExtension
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01101]	Size-constrained allocation of memory
[TPS_MANI_01102]	Specification of a namespace for an ImplementationDataType of category VECTOR
[TPS_MANI_01103]	Three-level approach to REST modeling
[TPS_MANI_01105]	Semantics of RestServiceInterface

▽

△

Number	Heading
[TPS_MANI_01106]	Specification of capabilities for the receiver of events or field notifiers
[TPS_MANI_01107]	Specification of capabilities for the sender of events or field notifiers
[TPS_MANI_01108]	Specification of capabilities for the caller of a methods or field setter/getter
[TPS_MANI_01109]	Semantics of UploadablePackageElement
[TPS_MANI_01110]	Semantics of SoftwareCluster
[TPS_MANI_01111]	Diagnostic Address of a SoftwareCluster
[TPS_MANI_01112]	Semantics of SoftwareClusterDesign
[TPS_MANI_01113]	Semantics of SoftwareClusterDesign.diagnosticAddress
[TPS_MANI_01114]	Relation of DiagnosticContributionSet to SoftwareCluster
[TPS_MANI_01115]	Specification of executable software within SoftwareCluster
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01117]	Semantics of SoftwareClusterDesign.intendedTargetMachine
[TPS_MANI_01118]	Relation between SoftwareClusterDesign and DiagnosticContributionSet
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterDesign
[TPS_MANI_01120]	Recursive definition of RestResourceDef
[TPS_MANI_01121]	Semantics of RestResourceDef.endpoint
[TPS_MANI_01122]	Arguments to endpoints
[TPS_MANI_01123]	System Triggered Event
[TPS_MANI_01124]	Semantics of RestElementDef
[TPS_MANI_01125]	Properties of REST elements can either be primitive or have array semantics
[TPS_MANI_01126]	Definition of string properties
[TPS_MANI_01127]	Limited support for data semantics in RestAbstractNumericalPropertyDef
[TPS_MANI_01128]	Difference between RestIntegerPropertyDef and RestNumberPropertyDef
[TPS_MANI_01129]	RestObjectRef is only needed for specific implementations of REST-based communication
[TPS_MANI_01130]	Structure of a typical URI for a REST service
[TPS_MANI_01131]	Impact of nested REST resources on the structure of REST URI
[TPS_MANI_01132]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03110]	Allowed components in system description with category category SOFTWARE_COMPONENT_SYSTEM_DESIGN_DESCRIPTION.
[TPS_MANI_03111]	Mapping between method and operation
[TPS_MANI_03112]	Mapping between an event and a dataElement
[TPS_MANI_03113]	Mapping between a field and elements of Classic Platform PortInterfaces

▽

△

Number	Heading
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03115]	Mapping between a fire and forget method and elements of Classic Platform PortInterfaces
[TPS_MANI_03116]	Size of a length field for a chosen string
[TPS_MANI_03117]	Default size for all string length fields
[TPS_MANI_03118]	Semantics of ServiceInterface.method with fireAndForget set to true
[TPS_MANI_03119]	Default value for the attribute fireAndForget of meta-class ClientServerOperation
[TPS_MANI_03120]	Signal-based ServiceInterface binding
[TPS_MANI_03121]	Signal-based VariableDataPrototype binding
[TPS_MANI_03122]	Signal-based Field binding
[TPS_MANI_03123]	Signal-based ClientServerOperation binding
[TPS_MANI_03124]	SignalBasedEventDeployment to ISignalTriggering mapping
[TPS_MANI_03125]	SignalBasedMethodDeployment to ISignalTriggerings mapping
[TPS_MANI_03126]	SignalBasedFieldDeployment to ISignalTriggerings mapping
[TPS_MANI_03127]	Usage of End2EndEventProtectionProps
[TPS_MANI_03128]	Usage of same dataId in case of Multi-Binding
[TPS_MANI_03129]	E2E profile
[TPS_MANI_03130]	Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03131]	Non-Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03132]	Semantics of E2E attributes in ReceiverComSpec
[TPS_MANI_03133]	Usage of ServiceInterfaceElementSecureComConfig
[TPS_MANI_03134]	Configuration of supported TLS ciphersuites
[TPS_MANI_03135]	Configuration of TLS PSK Identity
[TPS_MANI_03136]	Configuration of requirements for the TLS cryptographic job
[TPS_MANI_03137]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are not relevant in case of TLS communication
[TPS_MANI_03138]	SecOC Security Profile
[TPS_MANI_03139]	Standardized SecOC Security Profiles
[TPS_MANI_03140]	Non-Standardized SecOC Security Profiles
[TPS_MANI_03141]	Mapping between SecOcJobRequirement and CryptoJob
[TPS_MANI_03142]	Mapping between TlsJobRequirement and CryptoJob
[TPS_MANI_03143]	Mapping between PresharedKeyIdentity and CryptoKeySlot
[TPS_MANI_03144]	C++ language binding of ImplementationDataTypes of category STRING
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03146]	Configuration of timeouts for a selected machine state or function group state

▽

△

Number	Heading
[TPS_MANI_03147]	Mapping of a Process to a Machine
[TPS_MANI_03148]	Description of Core affinity
[TPS_MANI_03149]	Definition of a start-up timeout for a Process
[TPS_MANI_03150]	Definition of a termination timeout for a Process
[TPS_MANI_03151]	Default value for termination timeout
[TPS_MANI_03152]	Assignment of a StateDependentStartupConfig to a function group state
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_03500]	Definition of platform health management checkpoints
[TPS_MANI_03501]	Definition of platform health management supervised entities
[TPS_MANI_03502]	Enabling of PlatformHealthManagementContribution on a Machine
[TPS_MANI_03503]	Applicability of supervision to a specific Process
[TPS_MANI_03504]	Existence of SupervisionEntity
[TPS_MANI_03505]	Existence of PhmCheckpoint
[TPS_MANI_03506]	Optionality of SupervisionEntity and PhmCheckpoint
[TPS_MANI_03508]	Definition of an AliveSupervision for a PhmCheckpoint
[TPS_MANI_03509]	Definition of a CheckpointTransition
[TPS_MANI_03510]	Definition of LogicalSupervision
[TPS_MANI_03511]	Definition of DeadlineSupervision
[TPS_MANI_03512]	Applicability of global supervision to a specific Process
[TPS_MANI_03513]	Collection of SupervisionEntitys into a global supervision
[TPS_MANI_03514]	Expiration tolerance for GlobalSupervisionEntity
[TPS_MANI_03515]	Expiration tolerance for SupervisionEntity
[TPS_MANI_03516]	Condition evaluation for HealthChannelSupervision
[TPS_MANI_03517]	Condition evaluation for HealthChannelExternalMode
[TPS_MANI_03518]	LogicalExpression definition
[TPS_MANI_03519]	Rule definition
[TPS_MANI_03520]	Execution of PhmActionList with actionListExecution=triggeredOnEvaluation
[TPS_MANI_03521]	Execution of PhmActionList with actionListExecution=triggeredOnChange
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog

Table D.3: Added Traceables in 17-10

D.2.2 Changed Traceables in 17-10

Number	Heading
[TPS_MANI_01004]	Semantics of ServiceInterface.namespace
[TPS_MANI_01006]	Ordered definition of ServiceInterface.namespace
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01018]	ImplementationDataType of category VECTOR
[TPS_MANI_01030]	ImplementationDataType of category STRING
[TPS_MANI_03000]	Mapping of AdaptivePlatformServiceInstance to PortPrototypes
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03009]	Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03018]	Usage of SomeipProvidedEventGroup.multicastThreshold
[TPS_MANI_03023]	Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03024]	Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03049]	Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array length fields
[TPS_MANI_03104]	Default size for all structure length fields
[TPS_MANI_03105]	Default size for all union length fields
[TPS_MANI_03106]	Default size for all union type selector fields
[TPS_MANI_03107]	Default alignment for all dynamic DataPrototypes
[TPS_MANI_03108]	Default Byte Order for all DataPrototypes
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface

Table D.4: Changed Traceables in 17-10

D.2.3 Deleted Traceables in 17-10

Number	Heading
[TPS_MANI_03100]	Transport layer independent TransportLayerIndependentInstanceIds

Table D.5: Deleted Traceables in 17-10

D.2.4 Added Constraints in 17-10

Number	Heading
[constr_1522]	Semantics of ClientServerOperation.possibleError
[constr_1524]	Standardized values of PersistencyFileProxyInterface.category
[constr_1525]	Standardized values of PersistencyFile.category
[constr_1526]	Values of PersistencyFileArray.file.category
[constr_1527]	ImplementationDataTypeElement finally referenced as the target element in the context of an ImplementationDataTypeElementInAutosarDataPrototypeRef
[constr_1528]	Definition of optionality for multiple DataPrototypes typed by the same AutosarDataType
[constr_1529]	Standardized values of CryptoNeed.category
[constr_1530]	Standardized values of CryptoPrimitive.algorithmFamily and CryptoKeySlot.algorithmFamily
[constr_1531]	Standardized values of CryptoPrimitive.algorithmMode
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1533]	Applicability of ImplementationDataTypeElementExtension
[constr_1534]	Existence of DiagnosticSoftwareClusterProps
[constr_1535]	Existence of DiagnosticSoftwareClusterProps in the context of a DiagnosticContributionSet
[constr_1536]	Definition of SoftwareCluster applies for a single Machine
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_1542]	No nested definition of SoftwareCluster
[constr_1543]	Only one physical address per SoftwareCluster
[constr_3366]	System category for a system description with Adaptive Platform components
[constr_3367]	FieldMapping.notifierDataElement reference
[constr_3368]	FieldMapping.getterOperation reference
[constr_3369]	FieldMapping.setterOperation reference
[constr_3370]	InterfaceMapping shall map all elements of a single ServiceInterface
[constr_3371]	Mutually exclusive existence of FireAndForgetMapping.dataElement reference and FireAndForgetMapping.trigger reference
[constr_3372]	Restriction in usage of ApSomeipTransformationProps.sizeOfStringLengthField
[constr_3374]	method with attribute fireAndForget set to true shall not have any inout or out arguments
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApplicationError
[constr_3376]	FireAndForgetMapping shall reference only fire and forget methods
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedFieldToISignalTriggeringMapping
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time

▽

△

Number	Heading
[constr_3387]	Compatibility of PortPrototypes of different ServiceInterfaces
[constr_3388]	Compatibility of events
[constr_3389]	Compatibility of methods
[constr_3390]	Compatibility of fields
[constr_3391]	ServiceInterfaceElementSecureComConfig references to ServiceInterfaceDeployment elements
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication
[constr_3393]	Usage of shallRunOn and shallNotRunOn references
[constr_3394]	Default value for start-up timeout on the Machine is not configurable
[constr_3395]	TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface
[constr_3396]	Number of Process.modeDependentStartupConfig that refer to the same functionGroupMode
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroupMode and to a machineMode
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups
[constr_3527]	LogicalExpression referenced by one PhmRule

Table D.6: Added Constraints in 17-10

D.2.5 Changed Constraints in 17-10

Number	Heading
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1490]	Allowed value of category for reference ProcessToMachineMapping.process.executable
[constr_3290]	Transport Protocol attributes defined for a ProvidedSomeipServiceInstance
[constr_3296]	Transport Protocol attributes defined for a RequiredSomeipServiceInstance
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3361]	Selective definition of serialization settings

Table D.7: Changed Constraints in 17-10

D.2.6 Deleted Constraints in 17-10

Number	Heading
[constr_3291]	SomeipServiceInstanceToMachineMapping.portConfig aggregation restriction
[constr_3358]	Usage of PortPrototype and TransportLayerIndependentInstanceId to define the same Service Instance is not allowed
[constr_3360]	RPortPrototypeProps are related only to TransportLayerIndependentInstanceIds representing a consumer Service Instance

Table D.8: Deleted Constraints in 17-10

D.3 Constraint and Specification Item History of this document according to AUTOSAR Release 18-03

D.3.1 Added Traceables in 18-03

Number	Heading
[TPS_MANI_01135]	Semantics of PersistencyKeyValueDatabaseInterface.dataTypeForSerialization
[TPS_MANI_01136]	AutosarDataPrototype is the target of the CompositionDataPrototypeRef
[TPS_MANI_01137]	Applicable use cases for CompositionDataPrototypeRef
[TPS_MANI_01138]	Semantics of PersistencyKeyValueDatabaseInterface.dataElement
[TPS_MANI_01139]	Semantics of PersistencyKeyValueDatabaseInterface.updateStrategy
[TPS_MANI_01140]	Semantics of PersistencyDataElement.updateStrategy
[TPS_MANI_01141]	Semantics of PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01142]	Semantics of PersistencyFileProxy
[TPS_MANI_01143]	Semantics of PersistencyFileProxy.updateStrategy
[TPS_MANI_01144]	Semantics of PersistencyKeyValuePair
[TPS_MANI_01146]	Initial value for PersistencyKeyValuePair
[TPS_MANI_01147]	Semantics of PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01148]	Semantics of PersistencyKeyValuePair.updateStrategy
[TPS_MANI_01149]	Semantics of PersistencyFileArray.file
[TPS_MANI_01150]	Semantics of PersistencyFileArray
[TPS_MANI_01151]	Semantics of PersistencyFileArray.updateStrategy
[TPS_MANI_01152]	Semantics of PersistencyFile.updateStrategy
[TPS_MANI_01154]	PersistencyFileArray.updateStrategy overrides PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01155]	PersistencyKeyValueDatabase.updateStrategy overrides PersistencyKeyValueDatabaseInterface.updateStrategy

▽

△

Number	Heading
[TPS_MANI_01156]	PersistencyKeyValuePair.updateStrategy overrides PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01157]	Semantics of updateStrategy on collection level
[TPS_MANI_01158]	PersistencyFile.updateStrategy overrides PersistencyFileArray.updateStrategy
[TPS_MANI_01159]	Semantics of updateStrategy on element level
[TPS_MANI_01160]	Definition of initial value for PersistencyDataElement
[TPS_MANI_01161]	Impact of values of category on the semantics of SoftwareClusterDesign
[TPS_MANI_01162]	Semantics of SoftwareClusterDesign.dependsOn
[TPS_MANI_01163]	Impact of values of category on the semantics of SoftwareCluster
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01165]	Standardized value of UserDefinedServiceInterfaceDeployment.category
[TPS_MANI_01166]	Semantics of CppImplementationDataType
[TPS_MANI_01167]	AbstractImplementationDataType
[TPS_MANI_01168]	Specification of a namespace for a CppImplementationDataType
[TPS_MANI_01169]	Support for template data types
[TPS_MANI_01170]	Semantics of CppTemplateArgument.isVariadicTemplate
[TPS_MANI_01171]	Modeling of structured data types
[TPS_MANI_01172]	Description of type references in the scope of CppImplementationDataType
[TPS_MANI_01173]	Description of type references in the scope of CppImplementationDataTypeElement
[TPS_MANI_01174]	Semantics of reference in the role CppTemplateArgument.templateType
[TPS_MANI_01175]	Semantics of reference in the role CppTemplateArgument.allocator
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01178]	Semantics of RestHttpPortPrototypeMapping.acceptsEncoding
[TPS_MANI_01179]	Semantics of PersistencyFileProxy.contentUri/PersistencyFile.contentUri vs. PersistencyFileArray.uri and PersistencyFileProxy.fileName/PersistencyFile.fileName
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_01181]	Use cases for the application of DiagnosticServiceSwMapping
[TPS_MANI_01182]	PersistencyKeyValuePair.updateStrategy overrides PersistencyDataElement.updateStrategy
[TPS_MANI_01183]	PersistencyFile.updateStrategy overrides PersistencyFileProxy.updateStrategy
[TPS_MANI_03154]	ProvidedSomeipServiceInstance related configuration settings for events

▽

△

Number	Heading
[TPS_MANI_03155]	ProvidedSomeipServiceInstance related configuration settings for methods
[TPS_MANI_03156]	RequiredSomeipServiceInstance related configuration settings for methods
[TPS_MANI_03157]	Enabling of data accumulation for upd data transmission
[TPS_MANI_03158]	Configuration of a data accumulation on a ProvidedServiceInstance for transmission over udp
[TPS_MANI_03159]	Configuration of a data accumulation on a RequiredSomeipServiceInstance for transmission over udp
[TPS_MANI_03160]	Log and Trace configuration options in the Application Manifest
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03162]	Machine-specific configuration settings for the Log and Trace functional cluster
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03164]	Machine-specific configuration settings for DoIP
[TPS_MANI_03165]	Network configuration for DoIP
[TPS_MANI_03166]	Machine-specific configuration settings for NM module
[TPS_MANI_03167]	Network configuration for Nm
[TPS_MANI_03168]	Configuration of the SOME/IP load balancing option
[TPS_MANI_03169]	CppImplementationDataType with fixed size array semantics
[TPS_MANI_03170]	CppImplementationDataType of category ARRAY
[TPS_MANI_03171]	Value type of a CppImplementationDataType of category ARRAY
[TPS_MANI_03172]	Size of a CppImplementationDataType of category ARRAY
[TPS_MANI_03173]	multidimensional Array
[TPS_MANI_03174]	CppImplementationDataType with variable size array semantics
[TPS_MANI_03175]	CppImplementationDataType of category VECTOR
[TPS_MANI_03176]	Value type of a CppImplementationDataType of category VECTOR
[TPS_MANI_03177]	multidimensional Vector
[TPS_MANI_03178]	CppImplementationDataType of category STRING
[TPS_MANI_03179]	C++ language binding of CppImplementationDataTypes of category STRING
[TPS_MANI_03180]	Definition of Structures
[TPS_MANI_03181]	Definition of members in CppImplementationDataType of category STRUCTURE
[TPS_MANI_03182]	Definition of members in CppImplementationDataTypeElement of category STRUCTURE
[TPS_MANI_03183]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03184]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03185]	Structure of an CppImplementationDataType of category ASSOCIATIVE_MAP

▽

△

Number	Heading
[TPS_MANI_03186]	Usage of arraySize in case of a Vector
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03188]	Usage of an Allocator for a CppImplementationDataType of category STRING
[TPS_MANI_03189]	Definition of CppImplementationDataType of category VARIANT
[TPS_MANI_03190]	CppImplementationDataType of category VARIANT
[TPS_MANI_03191]	Definition of type alternatives stored in a VARIANT
[TPS_MANI_03192]	CppImplementationDataType of category VALUE
[TPS_MANI_03193]	CppImplementationDataType or CppImplementationDataTypeElement of category TYPE_REFERENCE
[TPS_MANI_03194]	Function Group State
[TPS_MANI_03195]	Off state in Function Group
[TPS_MANI_03196]	Semantics of CppImplementationDataTypeElementQualifier.anonymous attribute
[TPS_MANI_03525]	DDS ServiceInterface binding
[TPS_MANI_03526]	DDS VariableDataPrototype binding
[TPS_MANI_03527]	Definition of ProvidedDdsServiceInstance
[TPS_MANI_03528]	Definition of ProvidedDdsEventQosProps
[TPS_MANI_03529]	Definition of RequiredDdsServiceInstance
[TPS_MANI_03530]	Definition of RequiredDdsEventQosProps
[TPS_MANI_03531]	qosProfile of ProvidedDdsEventQosProps is optional
[TPS_MANI_03532]	qosProfile of RequiredDdsEventQosProps is optional
[TPS_MANI_03533]	DdsServiceInstanceToMachineMapping
[TPS_MANI_03534]	Definition of Platform Health Management Health Channel
[TPS_MANI_03535]	Definition of Time Synchronization interaction
[TPS_MANI_03536]	Time Synchronization interaction in a master role
[TPS_MANI_03537]	Time Synchronization interaction in a slave role
[TPS_MANI_03538]	Time Synchronization interaction with a local Time Base
[TPS_MANI_03539]	Definition of Time Bases
[TPS_MANI_03540]	Definition of PureLocalTimeBase
[TPS_MANI_03541]	Definition of SynchronizedSlaveTimeBase
[TPS_MANI_03542]	Definition of SynchronizedMasterTimeBase
[TPS_MANI_03543]	Definition of time sync correction attributes
[TPS_MANI_03544]	Definition of PlatformHealthManagementContribution
[TPS_MANI_03545]	Existence of HealthChannelExternalStatus
[TPS_MANI_03546]	Definition of reported health status RPortPrototype
[TPS_MANI_03547]	Definition of *offset* time domains
[TPS_MANI_03548]	Definition of TimeSyncPortPrototypeToTimeBaseMapping

▽

△

Number	Heading
[TPS_MANI_03549]	Usage of RPortPrototype for the interaction with Time Synchronization
[TPS_MANI_03550]	Usage of RPortPrototype for the interaction with Platform Health Management
[TPS_MANI_03551]	Definition of Time Base kind
[TPS_MANI_03552]	Supervision cycle for GlobalSupervision

Table D.9: Added Traceables in 18-03

D.3.2 Changed Traceables in 18-03

Number	Heading
[TPS_MANI_01006]	Ordered definition of ServiceInterface.namespace
[TPS_MANI_01008]	Semantics of ExecutableGroup
[TPS_MANI_01009]	Standardized values of ExecutableGroup.category
[TPS_MANI_01013]	Semantics of meta-class ModeDependentStartupConfig
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01042]	Definition of a linear ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping
[TPS_MANI_01068]	Semantics of PersistencyFileProxyInterface.maxNumberOfFiles
[TPS_MANI_01069]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping
[TPS_MANI_01080]	Semantics of PersistencyPortPrototypeToFileArrayMapping
[TPS_MANI_01097]	Assignment of TLV data ids for data structures with optional members
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01109]	Semantics of UploadablePackageElement
[TPS_MANI_01112]	Semantics of SoftwareClusterDesign
[TPS_MANI_01113]	Semantics of SoftwareClusterDesign.diagnosticAddress
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01117]	Semantics of SoftwareClusterDesign.intendedTargetMachine
[TPS_MANI_01118]	Relation between SoftwareClusterDesign and DiagnosticContributionSet
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterDesign

▽

△

Number	Heading
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a MachineDesign
[TPS_MANI_03002]	IP configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03003]	ProvidedSomeipServiceInstance Fanout
[TPS_MANI_03022]	Context of RequiredSomeipServiceInstance
[TPS_MANI_03110]	Allowed components in system description with category SYSTEM_DESIGN_DESCRIPTION.
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03152]	Assignment of a ModeDependentStartupConfig to a function group state
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_03500]	Definition of Platform Health Management Supervision and Checkpoints
[TPS_MANI_03503]	Applicability of supervision to a specific Process
[TPS_MANI_03505]	Existence of SupervisionCheckpoint
[TPS_MANI_03506]	Optionality of SupervisionCheckpoint
[TPS_MANI_03508]	Definition of an AliveSupervision for a SupervisionCheckpoint
[TPS_MANI_03509]	Definition of a CheckpointTransition
[TPS_MANI_03510]	Definition of LogicalSupervision
[TPS_MANI_03512]	Applicability of global supervision to a specific Process
[TPS_MANI_03513]	Collection of LocalSupervisions into a global supervision
[TPS_MANI_03514]	Expiration tolerance for GlobalSupervision
[TPS_MANI_03515]	Expiration tolerance for LocalSupervision
[TPS_MANI_03516]	Condition evaluation for HealthChannelSupervision
[TPS_MANI_03517]	Condition evaluation for HealthChannelExternalStatus

Table D.10: Changed Traceables in 18-03

D.3.3 Deleted Traceables in 18-03

Number	Heading
[TPS_MANI_01031]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01045]	Process.modeDependentStartupConfig that does not refer to a ModeDeclaration
[TPS_MANI_01132]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_03019]	TTL for SubscribeEventGroupAck Entries
[TPS_MANI_03501]	Definition of platform health management supervised entities

▽

△

Number	Heading
[TPS_MANI_03504]	Existence of SupervisionEntity

Table D.11: Deleted Traceables in 18-03

D.3.4 Added Constraints in 18-03

Number	Heading
[constr_1546]	Existence of attributes of ServiceInterfaceSubElement
[constr_1547]	Reference from ImplementationDataTypeExtension to ImplementationDataType
[constr_1548]	Reference from ImplementationDataTypeElementExtension to ImplementationDataTypeElement
[constr_1549]	Value of ProcessorCore.coreId
[constr_1550]	Reference from Process to ProcessDesign
[constr_1551]	Existence of CompositionDataPrototypeRef.dataPrototype vs. CompositionDataPrototypeRef.elementInImplDatatype
[constr_1553]	Restriction for ProcessToMachineMapping
[constr_1554]	Restriction regarding PersistencyKeyValuePair.initValue
[constr_1555]	Restriction applicable for PersistencyPortPrototypeToKeyValueDatabaseMapping.portPrototype
[constr_1556]	Restriction applicable for PersistencyPortPrototypeToFileArrayMapping.portPrototype
[constr_1557]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1558]	Existence of SoftwareClusterDesign.diagnosticAddress
[constr_1559]	Existence of SoftwareClusterDesign.subSoftwareCluster
[constr_1560]	Usage of SoftwareClusterDesign.requiredARElement
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1562]	Existence of SoftwareClusterDesign.diagnosticContribution
[constr_1563]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1564]	Existence of SoftwareCluster.diagnosticAddress
[constr_1565]	Existence of SoftwareCluster.subSoftwareCluster
[constr_1566]	Usage of SoftwareCluster.containedARElement
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn.dependentSoftwareCluster
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1569]	Restriction for the scope of RestHttpPortPrototypeMapping.acceptsEncoding

▽

△

Number	Heading
[constr_1570]	Restriction for UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC
[constr_1571]	CppImplementationDataType is limited
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1574]	Number of CppTemplateArguments with isVariadicTemplate set to True
[constr_1575]	Position of CppTemplateArgument with isVariadicTemplate set to True
[constr_1576]	Existence of CppTemplateArgument.templateType vs. CppTemplateArgument.allocator
[constr_1577]	Specification of a nativeDeclaration for a CppImplementationDataType
[constr_1578]	applicable data categories
[constr_1579]	SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1580]	Restriction for the usage of RestHttpPortPrototypeMapping.acceptsEncoding
[constr_1581]	Value of fileProxy.fileName
[constr_1582]	PersistencyKeyValuePair.valueDataType shall match to ImplementationDataType for the corresponding PersistencyDataElement
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1586]	DiagnosticServiceSwMapping.category set to DATA_ELEMENT
[constr_1587]	DiagnosticServiceSwMapping.category set to DATA_IDENTIFIER
[constr_1588]	DiagnosticServiceSwMapping.category set to GENERIC_UDS_SERVICE
[constr_1589]	Value of file.fileName
[constr_3408]	Value range of SomeipEventDeployment.eventId
[constr_3409]	Value range of SomeipMethodDeployment.methodId
[constr_3410]	Value range of SomeipServiceInterfaceDeployment.serviceInterfaceId
[constr_3411]	eventMulticastUdpPort, ipv4MulticastIpAddress and ipv6MulticastIpAddress not relevant for RequiredSomeipServiceInstances
[constr_3412]	OsModuleInstantiation shall have at least one ResourceGroup
[constr_3413]	ModeDependentStartupConfig of a Process is mapped to exactly one ResourceGroup
[constr_3414]	Allowed usage of EthernetNetworkConfiguration attributes
[constr_3415]	Value range of loadBalancingPriority
[constr_3416]	Value range of loadBalancingWeight
[constr_3417]	UserDefinedEventDeployments aggregated by a UserDefinedFieldDeployment
[constr_3418]	UserDefinedMethodDeployments aggregated by a UserDefinedFieldDeployment
[constr_3419]	Allowed usage of EthernetNetworkConfiguration attributes

▽

△

Number	Heading
[constr_3420]	System category for a design description that has one single Adaptive Machine in scope
[constr_3421]	Fibex elements applicable for a MACHINE_DESIGN_EXTRACT
[constr_3422]	CppImplementationDataType of category STRING and SwBaseType
[constr_3423]	ModeDependentStartupConfig of a Process shall reference a functionGroupMode or machineMode
[constr_3424]	ModeDependentStartupConfig shall never reference the functionGroupMode Off
[constr_3425]	Restriction of DoIpInstantiations on a Machine
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file
[constr_3428]	Structure shall own at least one element
[constr_3429]	No allocator usage for CppImplementationDataTypes of category VARIANT
[constr_3432]	Allowed subElements for Structures
[constr_3433]	Aggregation of templateArguments for a ARRAY
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3528]	Value range of domainId
[constr_3529]	Value range of serviceInstanceId
[constr_3530]	Mandatory definition of checkpointId
[constr_3531]	Mandatory definition of healthChannelId
[constr_3532]	Mandatory definition of statusId
[constr_3536]	Mandatory definition of supervisedEntityId

Table D.12: Added Constraints in 18-03

D.3.5 Changed Constraints in 18-03

Number	Heading
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances

▽

△

Number	Heading
[constr_3310]	SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3320]	Aggregation of CommunicationConnector by MachineDesign
[constr_3350]	Consistent value of category for ExecutableGroups referencing an Executable
[constr_3366]	System category for a system design description with Adaptive Platform and Classic Platform content

Table D.13: Changed Constraints in 18-03

D.3.6 Deleted Constraints in 18-03

Number	Heading
[constr_1480]	Mutual existence of CompositionDataPrototypeRef.elementInImplDatatype vs. attributes of CompositionDataPrototypeRef.dataPrototype
[constr_1505]	Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration
[constr_1525]	Standardized values of PersistencyFile.category
[constr_1526]	Values of PersistencyFileArray.file.category
[constr_1533]	Applicability of ImplementationDataTypeElementExtension

Table D.14: Deleted Constraints in 18-03

D.4 Constraint and Specification Item History of this document according to AUTOSAR Release 18-10

D.4.1 Added Traceables in 18-10

Number	Heading
[TPS_MANI_01184]	Definition of optional elements on the level of ApplicationDataType
[TPS_MANI_01185]	Definition of optional elements on the level of CppImplementationDataType
[TPS_MANI_01186]	Definition of the applicable wire type
[TPS_MANI_01187]	Matching pairs of PersistencyFileProxy and PersistencyFile
[TPS_MANI_01188]	Semantics of attribute schedulingPriority
[TPS_MANI_01189]	Software Cluster and DiagnosticContributionSet.category
[TPS_MANI_01190]	Semantics of ApApplicationError
[TPS_MANI_01191]	Modeling of possible errors
[TPS_MANI_01192]	Semantics of ApApplicationErrorDomain

▽

△

Number	Heading
[TPS_MANI_01193]	Combination of ModeDependentStartupConfig.machineMode and ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_01194]	Semantics of PersistencyKeyValueDatabaseInterface.minimumSustainedSize
[TPS_MANI_01195]	Semantics of PersistencyFileProxyInterface.minimumSustainedSize
[TPS_MANI_01196]	Semantics of PersistencyKeyValueDatabase.minimumSustainedSize
[TPS_MANI_01197]	Semantics of PersistencyKeyValueDatabase.maximumAllowedSize
[TPS_MANI_01198]	Semantics of ApApplicationErrorSet
[TPS_MANI_01199]	Semantics of DeterministicClientResourceNeeds
[TPS_MANI_01200]	Semantics of meta-class DeterministicClientResource
[TPS_MANI_01201]	Standardized values for attribute CppTemplateArgument.category
[TPS_MANI_01202]	Semantics of reference SoftwareCluster.moduleInstantiation
[TPS_MANI_01203]	Semantics of DeterministicClient
[TPS_MANI_01204]	Specification of redundancy of persistent data
[TPS_MANI_01205]	Semantics of meta-class PersistencyDeployment
[TPS_MANI_01206]	Modeling of redundancy in the context of PersistencyDeployment
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyCrc.algorithmFamily
[TPS_MANI_01208]	Definition of envionment variables in the scope of a Machine
[TPS_MANI_01209]	Definition of envionment variables in process scope
[TPS_MANI_01210]	Default encoding for all DataPrototypes typed by CppImplementationDataType of category STRING
[TPS_MANI_03197]	Semantics of StdCppImplementationDataType
[TPS_MANI_03198]	Semantics of CustomCppImplementationDataType
[TPS_MANI_03199]	Endpoint protection by SecureComProps
[TPS_MANI_03200]	SecureComProps for udp, tcp and multicast communication
[TPS_MANI_03201]	Semantics of CppTemplateArgument.inplace attribute
[TPS_MANI_03202]	Definition of bitfield types
[TPS_MANI_03203]	Configuration of IPsec
[TPS_MANI_03204]	Definition of IPSecRules
[TPS_MANI_03205]	IPsec connection type
[TPS_MANI_03206]	IPsec AH and ESP protocol configuration
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03208]	Protection of AdaptivePlatformServiceInstance by IPsec
[TPS_MANI_03209]	The meaning of MachineDesign.accessControl
[TPS_MANI_03210]	Specification of event specific communication attributes
[TPS_MANI_03211]	Specification of field specific communication attributes

▽

△

Number	Heading
[TPS_MANI_03212]	Specification of initial value for a field
[TPS_MANI_03213]	Semantics of meta-class TlsSecureComProps
[TPS_MANI_03214]	Existence of TlsCryptoCipherSuite.keyExchange vs. TlsSecureComProps.keyExchange
[TPS_MANI_03215]	Semantics of CryptoServiceCertificate
[TPS_MANI_03216]	Existence of TlsCryptoCipherSuite.certificate in the *client* role
[TPS_MANI_03217]	On-the-wire encoding for a chosen string
[TPS_MANI_03218]	Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration
[TPS_MANI_03219]	Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration
[TPS_MANI_03220]	Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration
[TPS_MANI_03221]	Default value for the attribute vehicleAnnouncementInterval of meta-class DoIpNetworkConfiguration
[TPS_MANI_03222]	Default value for the attribute tcpAliveCheckResponseTimeout of meta-class DoIpNetworkConfiguration
[TPS_MANI_03553]	Applicability of health channel to a specific Process
[TPS_MANI_03554]	Several SomeipServiceInstanceToMachineMappings with equal settings
[TPS_MANI_03555]	Mix of SomeipServiceInstanceToMachineMapping and signal-based communication
[TPS_MANI_03556]	DDS-RPC Service Binding
[TPS_MANI_03557]	DDS ClientServerOperation Binding
[TPS_MANI_03558]	DDS Field Binding
[TPS_MANI_03559]	Definition of DdsProvidedServiceInstance.methodQosProps
[TPS_MANI_03560]	qosProfile of DdsProvidedServiceInstance.methodQosProps is optional
[TPS_MANI_03561]	Definition of DdsProvidedServiceInstance.fieldNotifierQosProps
[TPS_MANI_03562]	qosProfile of DdsProvidedServiceInstance.fieldNotifierQosProps is optional
[TPS_MANI_03563]	Definition of DdsProvidedServiceInstance.fieldGetSetQosProps
[TPS_MANI_03564]	qosProfile of DdsProvidedServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03565]	Definition of DdsRequiredServiceInstance.methodQosProps
[TPS_MANI_03566]	qosProfile of DdsRequiredServiceInstance.methodQosProps is optional
[TPS_MANI_03567]	Definition of DdsRequiredServiceInstance.fieldNotifierQosProps

▽

△

Number	Heading
[TPS_MANI_03568]	qosProfile of DdsRequiredServiceInstance.fieldNotifierQosProps is optional
[TPS_MANI_03569]	Definition of DdsRequiredServiceInstance.fieldGetSetQosProps
[TPS_MANI_03570]	qosProfile of DdsRequiredServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03571]	transportPlugin for DdsProvidedServiceInstance
[TPS_MANI_03572]	transportPlugin for DdsRequiredServiceInstance

Table D.15: Added Traceables in 18-10

D.4.2 Changed Traceables in 18-10

Number	Heading
[TPS_MANI_01001]	Meaning of ServiceInterface
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01097]	Assignment of TLV data ids
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01147]	Semantics of PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01151]	Semantics of PersistencyFileArray.updateStrategy
[TPS_MANI_01166]	Semantics of CppImplementationDataType
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a MachineDesign
[TPS_MANI_03011]	Server Timing configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03021]	Requirements on the searched minor version from the client's point of view
[TPS_MANI_03025]	Client Timing configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03070]	Size of a length field for a chosen array or map
[TPS_MANI_03103]	Default size for all array and map length fields
[TPS_MANI_03124]	ServiceInterface.event to ISignalTriggering mapping
[TPS_MANI_03125]	ServiceInterface.method to ISignalTriggerings mapping
[TPS_MANI_03126]	ServiceInterface.field mapping to ISignalTriggerings
[TPS_MANI_03134]	Configuration of supported TLS ciphersuites
[TPS_MANI_03137]	ServiceInterfaceElementSecureComConfig is not relevant in case of TLS communication
[TPS_MANI_03157]	Enabling of data collection for upd data transmission

▽

△

Number	Heading
[TPS_MANI_03158]	Configuration of a data collection on a ProvidedServiceInstance for transmission over udp
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03170]	CppImplementationDataType of category ARRAY
[TPS_MANI_03173]	Definition of a multidimensional Array
[TPS_MANI_03175]	CppImplementationDataType of category VECTOR
[TPS_MANI_03177]	Definition of a multidimensional Vector
[TPS_MANI_03178]	StdCppImplementationDataType of category STRING
[TPS_MANI_03179]	C++ language binding of StdCppImplementationDataTypes of category STRING
[TPS_MANI_03180]	Definition of Structures
[TPS_MANI_03181]	Definition of members in StdCppImplementationDataType of category STRUCTURE
[TPS_MANI_03184]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03185]	Structure of an CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03193]	CppImplementationDataType of category TYPE_REFERENCE
[TPS_MANI_03196]	Semantics of CppImplementationDataTypeElementQualifier.inplace attribute
[TPS_MANI_03503]	Applicability of checkpoints to a specific Process
[TPS_MANI_03512]	Applicability of global supervision without Process context
[TPS_MANI_03516]	Condition evaluation for HealthChannelSupervision
[TPS_MANI_03518]	PhmLogicalExpression definition
[TPS_MANI_03519]	PhmRule definition
[TPS_MANI_03520]	Execution of PhmActionList with actionListExecution=triggeredOnEvaluation
[TPS_MANI_03521]	Execution of PhmActionList with actionListExecution=triggeredOnChange
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog
[TPS_MANI_03526]	DDS VariableDataPrototype binding
[TPS_MANI_03527]	Definition of DdsProvidedServiceInstance
[TPS_MANI_03528]	Definition of DdsProvidedServiceInstance.eventQosProps
[TPS_MANI_03529]	Definition of DdsRequiredServiceInstance
[TPS_MANI_03530]	Definition of DdsRequiredServiceInstance.eventQosProps
[TPS_MANI_03531]	qosProfile of DdsProvidedServiceInstance.eventQosProps is optional

▽

△

Number	Heading
[TPS_MANI_03532]	qosProfile of DdsRequiredServiceInstance.eventQosProps is optional
[TPS_MANI_03533]	DdsServiceInstanceToMachineMapping
[TPS_MANI_03552]	Supervision cycle for GlobalSupervision

Table D.16: Changed Traceables in 18-10

D.4.3 Deleted Traceables in 18-10

Number	Heading
[TPS_MANI_01008]	Semantics of ExecutableGroup
[TPS_MANI_01009]	Standardized values of ExecutableGroup.category
[TPS_MANI_01018]	ImplementationDataType of category VECTOR
[TPS_MANI_01028]	ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01029]	Usage of ImplementationDataType
[TPS_MANI_01030]	ImplementationDataType of category STRING
[TPS_MANI_01042]	Definition of a linear ImplementationDataType of category VECTOR
[TPS_MANI_01043]	Definition of a rectangular ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01055]	Definition of application-level errors
[TPS_MANI_01056]	Semantics of ApplicationError.errorContext
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_01074]	Specification of encryption of persistent data
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01077]	Specification of file encryption
[TPS_MANI_01082]	Eligibility of DataPrototypes for the definition of optionality
[TPS_MANI_01083]	Optionality is supported for ApplicationDataType as well as ImplementationDataType
[TPS_MANI_01084]	Optionality for a DataPrototype typed by an ApplicationDataType
[TPS_MANI_01085]	Definition of optionality for a DataPrototype typed by an ImplementationDataType
[TPS_MANI_01087]	Interaction with crypto software
[TPS_MANI_01088]	Semantics of CryptoNeed
[TPS_MANI_01089]	Relation between CryptoNeed and PortPrototype

▽

△

Number	Heading
[TPS_MANI_01090]	Modeling of crypto software as a platform module
[TPS_MANI_01091]	Semantics of CryptoJob
[TPS_MANI_01092]	Mapping between CryptoNeed and CryptoJob
[TPS_MANI_01093]	Semantics of CryptoDriver
[TPS_MANI_01094]	Scope of CryptoDriver
[TPS_MANI_01095]	Semantics of CryptoKeySlot
[TPS_MANI_01096]	Semantics of the CryptoPrimitive
[TPS_MANI_01098]	Constraints on the definition of an ImplementationDataType of category VECTOR
[TPS_MANI_01099]	Semantics of ImplementationDataTypeElementExtension
[TPS_MANI_01101]	Size-constrained allocation of memory
[TPS_MANI_01102]	Specification of a namespace for an ImplementationDataType of category VECTOR
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03121]	Signal-based VariableDataPrototype binding
[TPS_MANI_03122]	Signal-based Field binding
[TPS_MANI_03123]	Signal-based ClientServerOperation binding
[TPS_MANI_03135]	Configuration of TLS PSK Identity
[TPS_MANI_03136]	Configuration of requirements for the TLS cryptographic job
[TPS_MANI_03141]	Mapping between SecOcJobRequirement and CryptoJob
[TPS_MANI_03142]	Mapping between TlsJobRequirement and CryptoJob
[TPS_MANI_03143]	Mapping between PresharedKeyIdentity and CryptoKeySlot
[TPS_MANI_03144]	C++ language binding of ImplementationDataTypes of category STRING
[TPS_MANI_03182]	Definition of members in CppImplementationDataTypeElement of category STRUCTURE

Table D.17: Deleted Traceables in 18-10

D.4.4 Added Constraints in 18-10

Number	Heading
[constr_1593]	Completeness of the existence of a set of TlvDataIdDefinition.tlvArguments
[constr_1594]	Consistent assignment of TLV data ids to ApplicationRecordDataType
[constr_1595]	Consistent assignment of TLV data ids to CppImplementationDataType or CppImplementationDataTypeElement
[constr_1596]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ArgumentDataPrototype

▽

△

Number	Heading
[constr_1597]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ApplicationRecordElement
[constr_1598]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement
[constr_1599]	TlvDataIdDefinition referencing ArgumentDataPrototype
[constr_1600]	TlvDataIdDefinition referencing ApplicationRecordElement
[constr_1601]	TlvDataIdDefinition referencing CppImplementationDataTypeElement
[constr_1603]	Completeness of the existence of a set of TlvDataIdDefinition.tlvRecordElements
[constr_1604]	Completeness of the existence of a set of TlvDataIdDefinition.tlvSubElements
[constr_1605]	Standardized values of attribute Executable.category
[constr_1606]	Processes with mutual ExecutionDependencys
[constr_1613]	File name of matching pairs of PersistencyFileProxy and PersistencyFile
[constr_1614]	Existence of attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.sessionHandling
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling
[constr_1618]	Ability to shut down
[constr_1619]	Ability to restart
[constr_1620]	Value of schedulingPriority if schedulingPolicy is set to schedulingPolicyFifo or schedulingPolicyRoundRobin
[constr_1621]	Value of schedulingPriority if StartupConfig.schedulingPolicy is set to schedulingPolicyOther
[constr_1625]	Existence of reference ApApplicationError.errorDomain
[constr_1627]	Supported value range for attribute ApApplicationErrorDomain.value
[constr_1628]	Definition of static length field sizes in case of TLV usage
[constr_1629]	Identical sizes of length fields in case of TLV usage
[constr_1630]	No definition of length field sizes on DataPrototype level in case of TLV usage
[constr_1658]	Number of DiagnosticTroubleCodeUdsToClearConditionGroupMapping elements per DiagnosticTroubleCodeUds
[constr_1659]	Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace
[constr_1660]	Restriction for the usage of CppTemplateArgument.inplace
[constr_1661]	Multiplicity of OsModuleInstantiation.resourceGroup
[constr_1663]	Standardized values of attribute DiagnosticServiceDataIdentifierPortMapping.category
[constr_1664]	Unique ApApplicationError.shortName
[constr_1665]	Unique ApApplicationError.errorCode
[constr_1666]	References from PersistencyPortPrototypeToKeyValueDatabaseMapping to PersistencyKeyValueDatabase

▽

△

Number	Heading
[constr_1667]	References from PersistencyPortPrototypeToFileArrayMapping to PersistencyFileArray
[constr_1668]	Allowed combinations of PersistencyRedundancyCrc.length and algorithmFamily
[constr_1673]	Existence of attributes hasGetter, hasSetter, and hasNotifier
[constr_1674]	Suported encoding of StdCppImplementationDataType of category STRING
[constr_1675]	Existence of attribute ApSomeipTransformationProps.stringEncoding
[constr_1676]	Consistency of references shallRunOn and shallNotRunOn
[constr_1677]	Mutual exclusive existence of references shallRunOn and shallNotRunOn
[constr_1678]	Allowed values for attribute ApSomeipTransformationProps.stringEncoding
[constr_3443]	Specification of a namespace for a StdCppImplementationDataType
[constr_3446]	CppTemplateArgument with allocator reference and the inplace flag
[constr_3447]	ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0
[constr_3462]	CppTemplateArgument.templateType reference to StdCppImplementationDataType of category STRUCTURE and the inplace flag
[constr_3485]	UDP endpoint using DTLS can only serve provided or required service instances exclusively
[constr_3486]	TCP endpoint using TLS can only serve provided or required service instances exclusively.
[constr_3487]	TCP endpoint can only serve provided or required service instances exclusively
[constr_3492]	DoIpInstantiation.logicalAddress shall be defined as member in the DoIpRequestConfiguration
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_3494]	Mandatory Machine States
[constr_3495]	Supported value range for attribute DoIpInstantiation.eid
[constr_3496]	Supported value range for attribute DoIpInstantiation.gid
[constr_3497]	Supported value range for attribute DoIpInstantiation.maxRequestBytes
[constr_3498]	Supported value range for attribute DoIpInstantiation.logicalAddress
[constr_3499]	Supported value range for attribute DoIpRequestConfiguration.startAddress
[constr_3537]	LocalSupervision referenced once in the context of a GlobalSupervision
[constr_3538]	Only one ServiceInstanceToMachineMapping per technology and CommunicationConnector
[constr_3539]	Only one AliveSupervision per SupervisionCheckpoint
[constr_3540]	SupervisionCheckpoint in supervision graph
[constr_3541]	qosProfile mandatory for DdsProvidedServiceInstance
[constr_3542]	qosProfile mandatory for DdsRequiredServiceInstance
[constr_3543]	At least one transportPlugin definition required for each DdsProvidedServiceInstance

▽

△

Number	Heading
[constr_3544]	At least one transportPlugin definition required for each DdsRequiredServiceInstance
[constr_5000]	Supported value range for attribute DoIpRequestConfiguration.endAddress
[constr_5001]	Usage of DoIpNetworkConfiguration.eidUseMac
[constr_5002]	Supported values of ServiceInstanceToMachineMapping.category
[constr_5003]	Existence of TlsCryptoCipherSuite.certificate in the *server* role
[constr_5004]	Mapping of a Process to a Machine is mandatory in the Execution Manifest

Table D.18: Added Constraints in 18-10

D.4.5 Changed Constraints in 18-10

Number	Heading
[constr_1490]	Allowed value of category for reference ProcessToMachineMapping.process.executable
[constr_1551]	Existence of DataPrototypeInServiceInterfaceRef.dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1582]	PersistencyKeyValuePair.valueDataType shall match to AbstractImplementationDataType for the corresponding PersistencyDataElement
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1589]	Value of file.fileName
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApApplicationError
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication
[constr_3433]	Aggregation of templateArguments for an ARRAY
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3527]	PhmLogicalExpression referenced by one PhmRule
[constr_3528]	Value range of domainId
[constr_3529]	Value range of serviceInstanceId

Table D.19: Changed Constraints in 18-10

D.4.6 Deleted Constraints in 18-10

Number	Heading
[constr_1474]	SwDataDefProps applicable to ImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	ImplementationDataType of category STRING is limited
[constr_1476]	ImplementationDataType of category VECTOR is limited
[constr_1477]	ImplementationDataType of category ASSOCIATIVE_MAP is limited
[constr_1479]	No support for certain values of ImplementationDataType.category
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1485]	No subElement for ImplementationDataType of category STRING
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1487]	Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP
[constr_1491]	Semantics of ServiceInterface.possibleError
[constr_1493]	ArgumentDataPrototype referenced in the role ApplicationError.errorContext
[constr_1495]	Initial value for field
[constr_1506]	ImplementationDataType of category VECTOR shall not define dynamicArraySizeProfile
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_1522]	Semantics of ClientServerOperation.possibleError
[constr_1527]	ImplementationDataTypeElement finally referenced as the target element in the context of an ImplementationDataTypeElementInAutosarDataPrototypeRef
[constr_1528]	Definition of optionality for multiple DataPrototypes typed by the same AutosarDataType
[constr_1529]	Standardized values of CryptoNeed.category
[constr_1530]	Standardized values of CryptoPrimitive.algorithmFamily and CryptoKeySlot.algorithmFamily
[constr_1531]	Standardized values of CryptoPrimitive.algorithmMode
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_1546]	Existence of attributes of ServiceInterfaceSubElement
[constr_1547]	Reference from ImplementationDataTypeExtension to ImplementationDataType
[constr_1548]	Reference from ImplementationDataTypeElementExtension to ImplementationDataTypeElement
[constr_1577]	Specification of a nativeDeclaration for a CppImplementationDataType
[constr_1587]	DiagnosticServiceSwMapping.category set to DATA_IDENTIFIER
[constr_1588]	DiagnosticServiceSwMapping.category set to GENERIC_UDS_SERVICE

▽

△

Number	Heading
[constr_3293]	Mandatory information of a RequiredSomeipServiceInstance
[constr_3303]	ANY not allowed for SomeipServiceInterfaceDeployment.serviceInterfaceVersion
[constr_3350]	Consistent value of category for ExecutableGroups referencing an Executable
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedFieldToISignalTriggeringMapping
[constr_3422]	CppImplementationDataType of category STRING and SwBaseType
[constr_3428]	Structure shall own at least one element
[constr_3432]	Allowed subElements for Structures

Table D.20: Deleted Constraints in 18-10

D.5 Constraint and Specification Item History of this document according to AUTOSAR Release 19-03

D.5.1 Added Traceables in 19-03

Number	Heading
[TPS_MANI_01211]	Specification of executable software within SoftwareClusterDesign
[TPS_MANI_01212]	Usage of attribute typeEmitter in the context of a CustomCppImplementationDataType
[TPS_MANI_01213]	Semantics of meta-class StrongRevisionLabelString
[TPS_MANI_01214]	Semantics of SoftwareCluster.conflictsTo
[TPS_MANI_01215]	Semantics of meta-class SoftwareClusterDependencyFormula
[TPS_MANI_01216]	Semantics of meta-class SoftwareClusterDependencyFormulaPart
[TPS_MANI_01217]	Semantics of metaclass SoftwareClusterDependencyCompareCondition
[TPS_MANI_01218]	Cryptographic signature of SoftwareCluster
[TPS_MANI_01219]	License of software in included SoftwareCluster
[TPS_MANI_01220]	Release notes of software in included SoftwareCluster
[TPS_MANI_01221]	Semantics of meta-class SoftwarePackage
[TPS_MANI_01222]	Cryptographic signature of SoftwarePackage
[TPS_MANI_01223]	Semantics of attribute SoftwarePackage.packagerId
[TPS_MANI_01224]	Actions taken after installation of a SoftwarePackage
[TPS_MANI_01225]	Actions taken during installation of a SoftwarePackage
[TPS_MANI_01226]	Machine-specific configuration settings for the UCM module
[TPS_MANI_01227]	Semantics of attribute UcmModuleInstantiation.identifier
[TPS_MANI_01228]	Semantics of meta-class ProcessDesign

▽

△

Number	Heading
[TPS_MANI_01229]	Pre-allocation of a given ProcessDesign on a specific MachineDesign
[TPS_MANI_01230]	Semantics of DiagnosticProvidedDataMapping
[TPS_MANI_01231]	GrantDesign references ProcessDesign
[TPS_MANI_01232]	Semantics of meta-class ComOfferServiceGrantDesign
[TPS_MANI_01233]	Semantics of meta-class ComFindServiceGrantDesign
[TPS_MANI_01234]	Semantics of ComFieldGrantDesign
[TPS_MANI_01235]	Semantics of ComEventGrantDesign
[TPS_MANI_01236]	Semantics of ComMethodGrantDesign
[TPS_MANI_01237]	Semantics of meta-class ComFieldGrant
[TPS_MANI_01238]	Semantics of meta-class ComMethodGrant
[TPS_MANI_01239]	Semantics of meta-class ComEventGrant
[TPS_MANI_01240]	Semantics of meta-class ComOfferServiceGrant
[TPS_MANI_01241]	Semantics of meta-class ComFindServiceGrant
[TPS_MANI_01242]	PortInterfaces used for communication with the AUTOSAR Diagnostic Manager
[TPS_MANI_01243]	Semantics of DiagnosticDataIdentifierInterface
[TPS_MANI_01244]	Semantics of DiagnosticDataElementInterface
[TPS_MANI_01245]	Semantics of DiagnosticDataIdentifierGenericInterface
[TPS_MANI_01246]	Semantics of DiagnosticMonitorInterface
[TPS_MANI_01247]	Semantics of DiagnosticDTCInformationInterface
[TPS_MANI_01248]	Semantics of DiagnosticEventInterface
[TPS_MANI_01249]	Semantics of DiagnosticConditionInterface
[TPS_MANI_01250]	Semantics of DiagnosticIndicatorInterface
[TPS_MANI_01251]	Semantics of DiagnosticSecurityLevelInterface
[TPS_MANI_01252]	Semantics of DiagnosticServiceValidationInterface
[TPS_MANI_01253]	Semantics of DiagnosticOperationCycleInterface
[TPS_MANI_01254]	Semantics of DiagnosticGenericUdsInterface
[TPS_MANI_01255]	Semantics of DiagnosticGenericUdsInterface
[TPS_MANI_01256]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticIndicatorInterface
[TPS_MANI_01257]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticConditionInterface
[TPS_MANI_01258]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticGenericUdsInterface
[TPS_MANI_01259]	Mapping of DiagnosticClearCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01260]	Mapping of DiagnosticIndicator to PortPrototype(s) on the *AUTOSAR adaptive platform*

▽

△

Number	Heading
[TPS_MANI_01261]	Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01262]	Mapping of DiagnosticSecurityLevel to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01264]	Mapping of DiagnosticServiceInstance to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01265]	Semantics of DiagnosticDownloadInterface and DiagnosticDownloadInterface
[TPS_MANI_01266]	Mapping of DiagnosticServiceInstance for upload/download to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_03223]	Existence of CppImplementationDataType
[TPS_MANI_03224]	Modeling of a Partial Network Cluster
[TPS_MANI_03225]	References to vlans in PncMapping
[TPS_MANI_03226]	Collection of partialNetworks and vlans in NmNetworkHandle

Table D.21: Added Traceables in 19-03

D.5.2 Changed Traceables in 19-03

Number	Heading
[TPS_MANI_01012]	Formal modeling of application startup behavior
[TPS_MANI_01013]	Semantics of meta-class StateDependentStartupConfig
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01046]	Semantics of StateDependentStartupConfig.functionGroupState
[TPS_MANI_01049]	Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of DiagnosticEnableCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01136]	AutosarDataPrototype is the target of the DataPrototypeInServiceInterfaceRef
[TPS_MANI_01137]	Applicable use cases for DataPrototypeInServiceInterfaceRef
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyCrc.algorithmFamily

▽

△

Number	Heading
[TPS_MANI_03070]	Size of a length field for a chosen array or map
[TPS_MANI_03071]	Size of a length field for a chosen structure
[TPS_MANI_03072]	Size of a length field for a chosen union
[TPS_MANI_03073]	Alignment of a dynamic DataPrototype
[TPS_MANI_03074]	Size of a type selector field for a chosen union
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03116]	Size of a length field for a chosen string
[TPS_MANI_03127]	Usage of End2EndEventProtectionProps
[TPS_MANI_03128]	Usage of same dataId in case of Multi-Binding
[TPS_MANI_03152]	Assignment of a StateDependentStartupConfig to a function group state
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03190]	CppImplementationDataType of category VARIANT
[TPS_MANI_03202]	Definition of bitfield types
[TPS_MANI_03217]	On-the-wire encoding for a chosen string

Table D.22: Changed Traceables in 19-03

D.5.3 Deleted Traceables in 19-03

Number	Heading
[TPS_MANI_01038]	Diagnostic software mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01170]	Semantics of CppTemplateArgument.isVariadicTemplate
[TPS_MANI_01181]	Use cases for the application of DiagnosticServiceSwMapping
[TPS_MANI_01193]	Combination of ModeDependentStartupConfig.machineMode and ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_03066]	Description of machine states
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroupMode

Table D.23: Deleted Traceables in 19-03

D.5.4 Added Constraints in 19-03

Number	Heading
[constr_1687]	Definition of machine state
[constr_1688]	StateDependentStartupConfig shall only refer to function group states of the same function group
[constr_1689]	Modeling of a startup dependency between different Processes

▽

△

Number	Heading
[constr_1690]	SoftwareCluster shall only be referenced by a single SoftwarePackage.
[constr_1691]	UcmModuleInstantiation.identifier shall be unique
[constr_1692]	Value of schedulingPriority
[constr_1693]	Relation of Executable, ProcessDesign, and Process
[constr_1695]	Semantics of a Grant depends on the existence of IamModuleInstantiation
[constr_1696]	ClientServerOperation aggregated by DiagnosticRoutineInterface
[constr_1697]	Restriction for ClientServerOperation aggregated by a DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface
[constr_1698]	Target SwcServiceDependency of DiagnosticClearConditionPortMapping.swcServiceDependencyInExecutable
[constr_1699]	Target SwcServiceDependency of DiagnosticIndicatorPortMapping.swcServiceDependencyInExecutable
[constr_1700]	Target SwcServiceDependency of DiagnosticMemoryDestinationPortMapping.swcServiceDependencyInExecutable
[constr_1701]	Target SwcServiceDependency of DiagnosticSecurityLevelPortMapping.swcServiceDependencyInExecutable
[constr_1702]	Target SwcServiceDependency of DiagnosticServiceDataIdentifierPortMapping.swcServiceDependencyInExecutable
[constr_1703]	Target SwcServiceDependency of DiagnosticGenericUdsPortMapping.swcServiceDependencyInExecutable
[constr_1704]	Target SwcServiceDependency of DiagnosticUploadDownloadPortMapping.swcServiceDependencyInExecutable
[constr_5033]	Compatibility of data types with category VALUE
[constr_5034]	Compatibility of data types with category BOOLEAN
[constr_5035]	Compatibility of data types with category STRING
[constr_5036]	Compatibility of data types with category ARRAY
[constr_5037]	Compatibility of data types with category ARRAY with variableSize
[constr_5038]	Compatibility of data types with category ARRAY with fixedSize
[constr_5039]	Compatibility of data types with category STRUCTURE
[constr_5040]	Compatibility of ApplicationRecordDataType and CppImplementationDataType that both represent an Optional Element Structure
[constr_5041]	Compatibility of data types with category ASSOCIATIVE_MAP
[constr_5042]	No data type mapping for CppImplementationDataType of category VARIANT
[constr_5043]	Forbidden mappings to CppImplementationDataType
[constr_5044]	DataTypeMap for composite data types
[constr_5045]	Only one SomeipServiceDiscovery configuration per VLAN is allowed
[constr_5046]	Usage of DoIpNetworkConfiguration.eidUseMac
[constr_5047]	Supported values of ServiceInstanceToMachineMapping.category
[constr_5048]	Existence of TlsCryptoCipherSuite.certificate in the *server* role

Table D.24: Added Constraints in 19-03

D.5.5 Changed Constraints in 19-03

Number	Heading
[constr_1481]	Usage of DataPrototypeInServiceInterfaceRef in the *AUTOSAR adaptive platform*
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable
[constr_1501]	Target SwcServiceDependency of DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable
[constr_1502]	Target SwcServiceDependency of DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable
[constr_1503]	Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable
[constr_1551]	Existence of DataPrototypeInServiceInterfaceRef.dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn/conflictsTo
[constr_1595]	Consistent assignment of TLV data ids to CppImplementationDataType or CppImplementationDataTypeElement
[constr_1606]	Processes with mutual ExecutionDependencys
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling
[constr_1618]	Ability to shut down
[constr_1619]	Ability to restart
[constr_3396]	Number of Process.stateDependentStartupConfig that refer to the same functionGroupState
[constr_3413]	StateDependentStartupConfig of a Process is mapped to exactly one ResourceGroup
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3423]	StateDependentStartupConfig of a Process shall reference a functionGroupState
[constr_3424]	StateDependentStartupConfig shall never reference the functionGroupState Off
[constr_3447]	ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0

Table D.25: Changed Constraints in 19-03

D.5.6 Deleted Constraints in 19-03

Number	Heading
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same machineMode

▽

△

Number	Heading
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1574]	Number of CppTemplateArguments with isVariadicTemplate set to True
[constr_1575]	Position of CppTemplateArgument with isVariadicTemplate set to True
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1586]	DiagnosticServiceSwMapping.category set to DATA_ELEMENT
[constr_1620]	Value of schedulingPriority if schedulingPolicy is set to scheduling-PolicyFifo or schedulingPolicyRoundRobin
[constr_1621]	Value of schedulingPriority if StartupConfig.schedulingPolicy is set to schedulingPolicyOther
[constr_1663]	Standardized values of attribute DiagnosticServiceDataIdentifierMap-ping.category
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroupMode and to a machineMode
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups
[constr_3494]	Mandatory Machine States
[constr_3531]	Mandatory definition of healthChannelId
[constr_3536]	Mandatory definition of supervisedEntityId
[constr_5001]	Usage of DoIpNetworkConfiguration.eidUseMac
[constr_5002]	Supported values of ServiceInstanceToMachineMapping.category
[constr_5003]	Existence of TlsCryptoCipherSuite.certificate in the *server* role

Table D.26: Deleted Constraints in 19-03

E Splitable Elements in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpSplitable≫ in the scope of this document.

Each entry in the table consists of the identification of the specific model element itself and the applicable value of the tagged value atp.Splitkey.

For more information about the concept of splitable model elements and how these shall be treated please refer to [6].

Name of splitable element	Splitkey
AdaptiveApplicationSwComponentType.internalBehavior	internalBehavior, variationPoint.shortLabel
DiagnosticClearConditionGroup.clearCondition	clearCondition
DiagnosticClearConditionPortMapping.process	process
DiagnosticGenericUdsPortMapping.process	process
DiagnosticIndicatorPortMapping.process	process
DiagnosticMemoryDestinationPortMapping.process	process
DiagnosticSecurityLevelPortMapping.process	process
DiagnosticServiceDataIdentifierPortMapping.process	process
DiagnosticUploadDownloadPortMapping.process	process
IamModuleInstantiation.grant	grant
InterfaceMappingSet.interfaceMapping	shortName,variationPoint.shortLabel
Machine.environmentVariable	environmentVariable
Machine.functionGroup	shortName, variationPoint.shortLabel
Machine.moduleInstantiation	shortName
Machine.perStateTimeout	perStateTimeout
Machine.secureCommunicationDeployment	shortName, variationPoint.shortLabel
PlatformHealthManagementContribution.action	shortName
PlatformHealthManagementContribution.arbitration	shortName
PlatformHealthManagementContribution.checkpoint	shortName
PlatformHealthManagementContribution.globalSupervision	shortName
PlatformHealthManagementContribution.healthChannel	shortName
PlatformHealthManagementContribution.localSupervision	shortName
SoftwareCluster.conflictsTo	conflictsTo
SoftwareCluster.containedARElement	shortName
SoftwareCluster.containedPackageElement	containedPackageElement
SoftwareCluster.dependsOn	dependsOn
SoftwareCluster.diagnosticAddress	diagnosticAddress
SoftwareCluster.moduleInstantiation	moduleInstantiation
SoftwareCluster.subSoftwareCluster	subSoftwareCluster
SoftwareClusterDesign.containedProcess	containedProcess
SoftwareClusterDesign.dependsOn	dependsOn
SoftwareClusterDesign.diagnosticAddress	diagnosticAddress
SoftwareClusterDesign.diagnosticContribution	diagnosticContribution
SoftwareClusterDesign.requiredARElement	requiredARElement
SoftwareClusterDesign.requiredFibexElement	requiredFibexElement
SoftwareClusterDesign.requiredPackageElement	requiredPackageElement
SoftwareClusterDesign.subSoftwareCluster	subSoftwareCluster
TlvDataIdDefinition.tlvSubElement	tlvSubElement

TransformationPropsToServiceInterfaceElementMapping.tlvDataId	tlvDataId

Table E.1: Usage of splitable elements

F Variation Points in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpVariation≫ in the scope of this document.

Each entry in the table consists of the identification of the model element itself and the applicable value of the tagged value vh.latestBindingTime.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [6].

Variation Point	Latest Binding Time
AdaptiveApplicationSwComponentType.internalBehavior	preCompileTime
CppImplementationDataType.arraySize	preCompileTime
InterfaceMappingSet.interfaceMapping	systemDesignTime
Machine.functionGroup	preCompileTime
ServiceInterface.event	blueprintDerivationTime
ServiceInterface.field	blueprintDerivationTime
ServiceInterface.method	blueprintDerivationTime

Table F.1: Usage of variation points

G Used classes in Manifest files

G.1 Used classes in Machine Manifest

Used Class	Base
AdaptiveModuleInstantiation	other
CommunicationConnector	other
DoIpInstantiation	other
DoIpNetworkConfiguration	other
DoIpRequestConfiguration	other
EnterExitTimeout	other
EthernetCluster	PackageableElement
EthernetCommunicationConnector	other
EthernetNetworkConfiguration	other
EthernetPhysicalChannel	other
GenericModuleInstantiation	other
LogAndTraceInstantiation	other
Machine	PackageableElement
MachineDesign	PackageableElement
MacMulticastGroup	other
ModeDeclaration	other
ModeDeclarationGroup	PackageableElement
ModeDeclarationGroupPrototype	other
NetworkConfiguration	other
NetworkEndpoint	other
NetworkEndpointAddress	other
NmCluster	other
NmConfig	PackageableElement
NmInstantiation	other
NmNode	other
NonOsModuleInstantiation	other
OsModuleInstantiation	other
PerStateTimeout	other
Processor	other
ProcessorCore	other
PskIdentityToKeySlotMapping	other
PureLocalTimeBase	other
ResourceGroup	other
SecOcDeployment	other
SecOcJobMapping	other
SecureCommunicationDeployment	other
ServiceDiscoveryConfiguration	other
SomeipServiceDiscovery	other
SynchronizedMasterTimeBase	other
SynchronizedSlaveTimeBase	other
TimeBaseResource	other
TimeSyncModuleInstantiation	other
TlsDeployment	other
TlsJobMapping	other
UdpNmCluster	other
UdpNmNode	other

Table G.1: Used classes in MachineManifest

G.2 Used classes in Execution Manifest

Used Class	Base
AliveSupervision	other
CheckpointTransition	other
DeadlineSupervision	other
ExecutionDependency	other
GlobalSupervision	other
HealthChannel	other
HealthChannelExternalStatus	other
HealthChannelSupervision	other
HttpAcceptEncoding	other
LocalSupervision	other
LogicalSupervision	other
ModeDeclaration	other
ModeDeclarationGroup	PackageableElement
ModeDeclarationGroupPrototype	other
PersistencyFile	PackageableElement
PersistencyFileArray	PackageableElement
PersistencyKeyValueDatabase	PackageableElement
PersistencyKeyValuePair	other
PersistencyPortPrototypeToFileArrayMapping	PackageableElement
PersistencyPortPrototypeToKeyValueDatabaseMapping	PackageableElement
PhmAction	other
PhmActionItem	other
PhmActionList	other
PhmArbitration	other
PhmContributionToMachineMapping	PackageableElement
PhmLogicalExpression	other
PhmRule	other
PlatformHealthManagementContribution	PackageableElement
PlatformPhmActionItem	other
Process	PackageableElement
ProcessPhmActionItem	other
ProcessToMachineMapping	other
ProcessToMachineMappingSet	PackageableElement
RestHttpPortPrototypeMapping	PackageableElement
ServiceInstanceToPortPrototypeMapping	PackageableElement
StartupConfig	other
StartupConfigSet	PackageableElement
StartupOption	other
StateDependentStartupConfig	other
SupervisionCheckpoint	other
WatchdogPhmActionItem	other

Table G.2: Used classes in ExecutionManifest

G.3 Used classes in Service Instance Manifest

Used Class	Base
AdaptivePlatformServiceInstance	PackageableElement
CryptoServiceCertificate	PackageableElement

CryptoServiceKey	PackageableElement
CryptoServicePrimitive	PackageableElement
DdsEventDeployment	other
DdsEventQosProps	other
DdsMethodQosProps	other
DdsProvidedServiceInstance	PackageableElement
DdsRequiredServiceInstance	PackageableElement
DdsServiceInstanceToMachineMapping	PackageableElement
DdsServiceInterfaceDeployment	PackageableElement
E2EProfileConfiguration	other
E2EProfileConfigurationSet	PackageableElement
End2EndEventProtectionProps	other
InitialSdDelayConfig	other
ProvidedApServiceInstance	PackageableElement
ProvidedSomeipServiceInstance	PackageableElement
ProvidedUserDefinedServiceInstance	PackageableElement
RequestResponseDelay	other
RequiredApServiceInstance	PackageableElement
RequiredSomeipServiceInstance	PackageableElement
RequiredUserDefinedServiceInstance	PackageableElement
SecOcJobRequirement	other
SecOcSecureComProps	other
SecureComProps	other
SecureComPropsSet	PackageableElement
ServiceEventDeployment	other
ServiceFieldDeployment	other
ServiceInstanceToMachineMapping	PackageableElement
ServiceInterfaceDeployment	PackageableElement
ServiceInterfaceElementSecureComConfig	other
ServiceMethodDeployment	other
SomeipCollectionProps	other
SomeipEventDeployment	other
SomeipEventGroup	other
SomeipEventProps	other
SomeipFieldDeployment	other
SomeipMethodDeployment	other
SomeipMethodProps	other
SomeipProvidedEventGroup	other
SomeipRequiredEventGroup	other
SomeipSdClientEventGroupTimingConfig	PackageableElement
SomeipSdClientServiceInstanceConfig	PackageableElement
SomeipSdServerEventGroupTimingConfig	PackageableElement
SomeipSdServerServiceInstanceConfig	PackageableElement
SomeipServiceInstanceToMachineMapping	PackageableElement
SomeipServiceInterfaceDeployment	PackageableElement
SomeipServiceInterfaceVersion	other
TagWithOptionalValue	other
TlsCryptoCipherSuite	other
TlsSecureComProps	other
UserDefinedEventDeployment	other
UserDefinedFieldDeployment	other
UserDefinedMethodDeployment	other
UserDefinedServiceInstanceToMachineMapping	PackageableElement

UserDefinedServiceInterfaceDeployment	PackageableElement

Table G.3: Used classes in ServiceInstanceManifest