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	17-10

Document Change History			
Date	**Release**	**Changed by**	**Description**
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

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] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[6] ISO/IEC 14882:2011, Information technology – Programming languages – C++
http://www.iso.org

[7] Specification of Communication Management
AUTOSAR_SWS_CommunicationManagement

[8] 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

[9] Specification of Crypto Interface for Adaptive Platform
AUTOSAR_SWS_AdaptiveCryptoInterface

[10] System Template
AUTOSAR_TPS_SystemTemplate

[11] SOME/IP Protocol Specification
AUTOSAR_PRS_SOMEIPProtocol

[12] Diagnostic Extract Template
AUTOSAR_TPS_DiagnosticExtractTemplate

[13] Specification of RESTful Communication for Adaptive Platform
AUTOSAR_SWS_REST

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

[15] Specification of Execution Management
AUTOSAR_SWS_ExecutionManagement

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

[17] Specification of ECU Resource Template
AUTOSAR_TPS_ECUResourceTemplate

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 remains on the design level and describes the modeling of communication with web services following the REST pattern

Section 6 represents that counterpart to section 3 on deployment level, it describes the content of the so-called *application manifest*.

Section 7 provides a detailed description of how service-oriented communication shall be configured on *manifest* level.

Section 8 describes the options for configuring a machine by means of a *manifest*.

Section 9 describes the big picture of *AUTOSAR classic platform* and *AUTOSAR adaptive platform* communicating via service-oriented communication.

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 is the first in a string of sections that explain the manifest content of platform module functionality, in case of section 11 the manifest content for persistency is described.

Section 12 describes the manifest content needed for the configuration of the crypto platform module.

Section 13 describes the manifest content needed for the configuration of the communication platform module with respect to secure communication.

[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.

Section 14 lays out the details of the configuration of the platform health management module.

Section 15 describes the idea behind and the configuration of the concept of an uploadable software package.

Section 16 describes the manifest content needed for the configuration of the communication platform module with respect to REST.

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
DES	Data Encryption Standard
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
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
NM	Network Management
NV	Non-Volatile
OEM	Original Equipment Manufacturer
OS	Operating System
PDU	Protocol Data Unit

Abbreviation	*Meaning*
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
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, variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
introductio n	Documentation Block	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_00001]	Adaptive AUTOSAR Application	[TPS_MANI_01008] [TPS_MANI_01009]
[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]

[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_01055] [TPS_MANI_01064] [TPS_MANI_03118] [TPS_MANI_03119]
[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_01038] [TPS_MANI_01048] [TPS_MANI_01049] [TPS_MANI_01050] [TPS_MANI_01051] [TPS_MANI_01060]
[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_01045] [TPS_MANI_01046] [TPS_MANI_01059] [TPS_MANI_01061] [TPS_MANI_03153]
[RS_MANI_00008]	Service interface deployment to a transport layer mechanism	[TPS_MANI_01132] [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]
[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]
[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_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]
[RS_MANI_00016]	Usage of data types specifically on the AUTOSAR adaptive platform	[TPS_MANI_01016] [TPS_MANI_01018] [TPS_MANI_01027] [TPS_MANI_01028] [TPS_MANI_01029] [TPS_MANI_01030] [TPS_MANI_01042] [TPS_MANI_01043] [TPS_MANI_01044] [TPS_MANI_01047] [TPS_MANI_01062] [TPS_MANI_01063] [TPS_MANI_01098] [TPS_MANI_01099] [TPS_MANI_01100] [TPS_MANI_01101] [TPS_MANI_01102] [TPS_MANI_03144]

[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] [TPS_MANI_01058]
[RS_MANI_00018]	Network connections of the machine	[TPS_MANI_03035] [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] [TPS_MANI_03066]
[RS_MANI_00022]	Adaptive Platform configuration	[TPS_MANI_03035]
[RS_MANI_00023]	Adaptive Module configuration	[TPS_MANI_03035] [TPS_MANI_03056] [TPS_MANI_03096] [TPS_MANI_03098] [TPS_MANI_03502] [TPS_MANI_03503] [TPS_MANI_03504] [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]
[RS_MANI_00024]	SOME/IP transport layer mechanisms	[TPS_MANI_01132] [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_03019] [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]

[RS_MANI_00025]	Definition and configuration of serialization	[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_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_01074] [TPS_MANI_01075] [TPS_MANI_01077] [TPS_MANI_01078] [TPS_MANI_01079] [TPS_MANI_01080] [TPS_MANI_01081]
[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_03121] [TPS_MANI_03122] [TPS_MANI_03123] [TPS_MANI_03124] [TPS_MANI_03125] [TPS_MANI_03126]
[RS_MANI_00030]	Definition of optional elements in composite data structures	[TPS_MANI_01082] [TPS_MANI_01083] [TPS_MANI_01084] [TPS_MANI_01085] [TPS_MANI_01097] [TPS_MANI_01133] [TPS_MANI_01134]
[RS_MANI_00031]	Interaction with Crypto Software	[TPS_MANI_01087] [TPS_MANI_01088] [TPS_MANI_01089] [TPS_MANI_01090] [TPS_MANI_01091] [TPS_MANI_01092] [TPS_MANI_01093] [TPS_MANI_01094] [TPS_MANI_01095] [TPS_MANI_01096] [TPS_MANI_03141] [TPS_MANI_03142] [TPS_MANI_03143]
[RS_MANI_00032]	Support for platform health management	[TPS_MANI_03500] [TPS_MANI_03501] [TPS_MANI_03502] [TPS_MANI_03503] [TPS_MANI_03504] [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]
[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]
[RS_MANI_00034]	Specification of capabilities	[TPS_MANI_01106] [TPS_MANI_01107] [TPS_MANI_01108]

[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]
[RS_MANI_00036]	Configuration of security protection	[TPS_MANI_03133] [TPS_MANI_03134] [TPS_MANI_03135] [TPS_MANI_03136] [TPS_MANI_03137] [TPS_MANI_03138] [TPS_MANI_03139] [TPS_MANI_03140] [TPS_MANI_03141] [TPS_MANI_03142] [TPS_MANI_03143]

Table 1.3: RequirementsTracing

2 Big Picture of Manifest Definition

2.1 Design vs. Deployment

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*.

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 and 5.

Chapters 6, 7, 8, 11, 12, 13, and 14 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] 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] 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 [5].

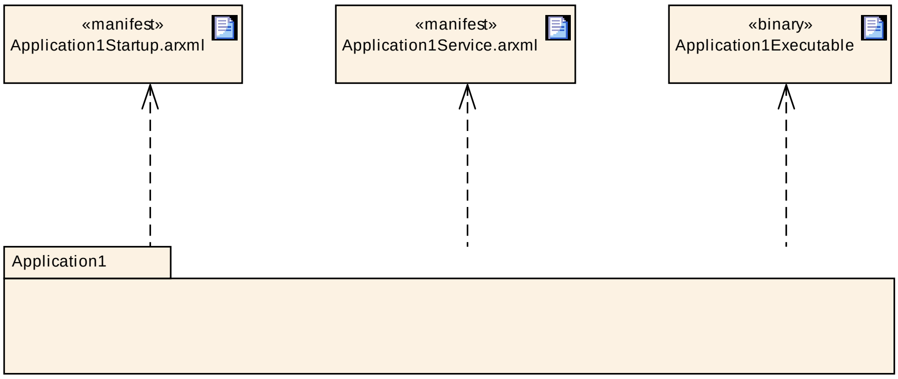

Figure 2.1: Example usage of several manifest files within one software delivery

[TPS_MANI_01021] 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**. ⌋*()*

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 [5].

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] 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, see [TPS_MANI_01009]) 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:

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

An Application 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 6.

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 7.

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 8.

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 3.7)
- Aspects of the fine-grained configuration of interaction with the "outside world" from the perspective of the inside of a software-component (section 3.8)
- Adaptive AUTOSAR application as a starting point for the transition towards the deployment (section 3.9)
- Configuration of transformation properties (section 3.11)

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 a 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=AdaptiveApplicationSw ComponentTypes			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
internalBe havior	AdaptiveSwcInt ernalBehavior	0..1	aggr	This aggregation represents the internal behavior of the AdaptiveApplicationSwComponentType for the AUTOSAR adaptive platform. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior, variation Point.shortLabel; atp.Status=draft vh.latestBindingTime=preCompileTime

Table 3.1: AdaptiveApplicationSwComponentType

Class	**AdaptiveSwcInternalBehavior**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::AdaptiveInternal Behavior			
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***

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 ApplicationDataTypes and ImplementationDataTypes.

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 Fields (see [TPS_MANI_01034]).

3.3.2 ApplicationDataType

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

In addition to the ApplicationDataTypes supported on the *AUTOSAR classic platform*, there are further ApplicationDataTypes 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 ApplicationDataTypes only.			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
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 the definition of ApplicationDataTypes on the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*, special regulations apply on the level of ImplementationDataTypes on the *AUTOSAR adaptive platform*.

For more information about the modeling of string data types on the level of ImplementationDataType please refer to section 3.3.3.1.

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] 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 cat-

`egory STRING` could rightfully be used **on both** the *AUTOSAR adaptive platform* and the *AUTOSAR classic platform*.

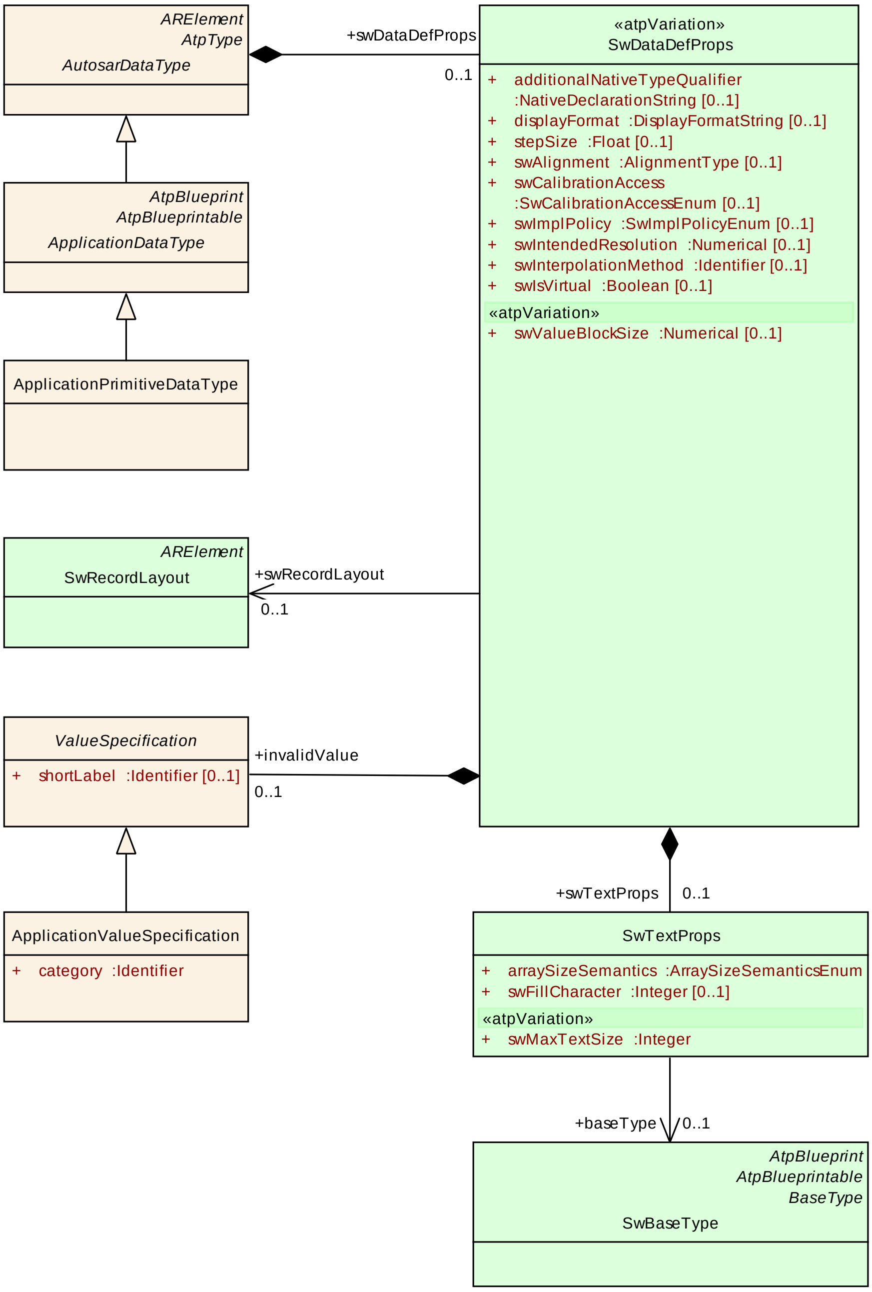

Figure 3.1: Specification of textual strings

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, Atp Classifier, AtpType, AutosarDataType, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
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*
arraySizeSemantics	ArraySizeSemanticsEnum	1	attr	This attribute controls the semantics of the arraysize for the array representing the string in an ImplementationDataType. It is there to support a safe conversion between ApplicationDatatype and ImplementationDatatype, even for variable length strings as required e.g. for Support of SAE J1939.
baseType	SwBaseType	0..1	ref	This is the base type of one character in the string. In particular this baseType denotes the intended encoding of the characters in the string on level of ApplicationDataType. **Tags:** xml.sequenceOffset=30
swFillCharacter	Integer	0..1	attr	Filler character for text parameter to pad up to the maximum length swMaxTextSize. The value will be interpreted according to the encoding specified in the associated base type of the data object, e.g. 0x30 (hex) represents the ASCII character zero as filler character and 0 (dec) represents an end of string as filler character. The usage of the fill character depends on the arraySizeSemantics. **Tags:** xml.sequenceOffset=40

Attribute	*Type*	*Mul.*	*Kind*	*Note*
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] 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 *value*s (formalized as `ApplicationAssocMapDataType.value` that are also in turn typed by an `ApplicationDataType`). ⌋*(RS_MANI_00016)*

[constr_3349] Usage of `ApplicationAssocMapDataType` is limited ⌈ The usage of an `ApplicationAssocMapDataType` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s 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] 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].

[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

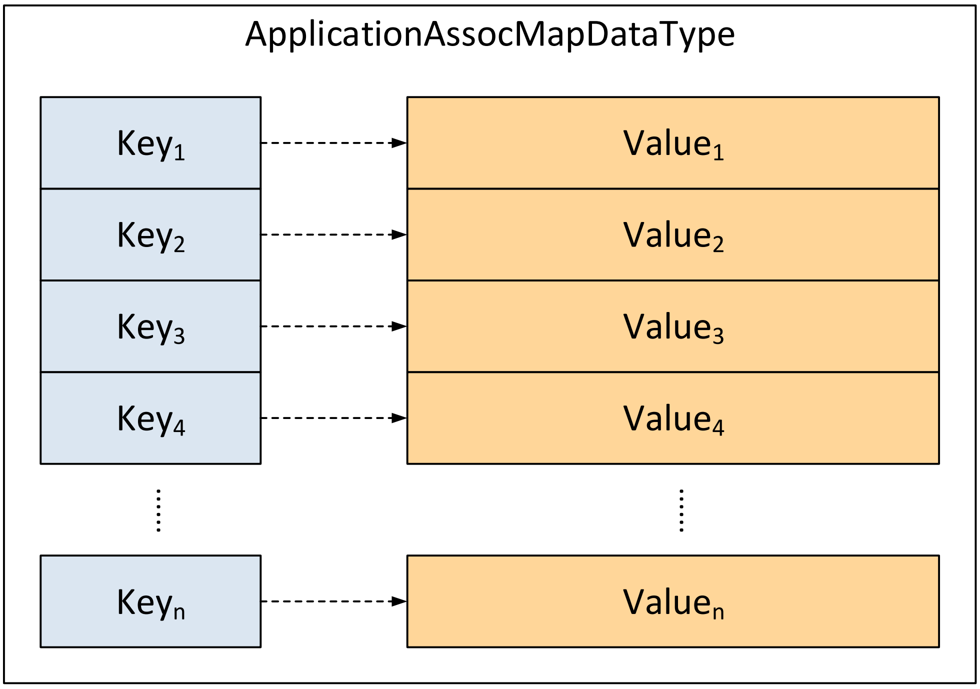

Figure 3.2: Example ApplicationAssocMapDataType on the *AUTOSAR adaptive platform*

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 `key`s are associated with `value`s 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 `std::map` (where the implementation is opaque to the client programmer).

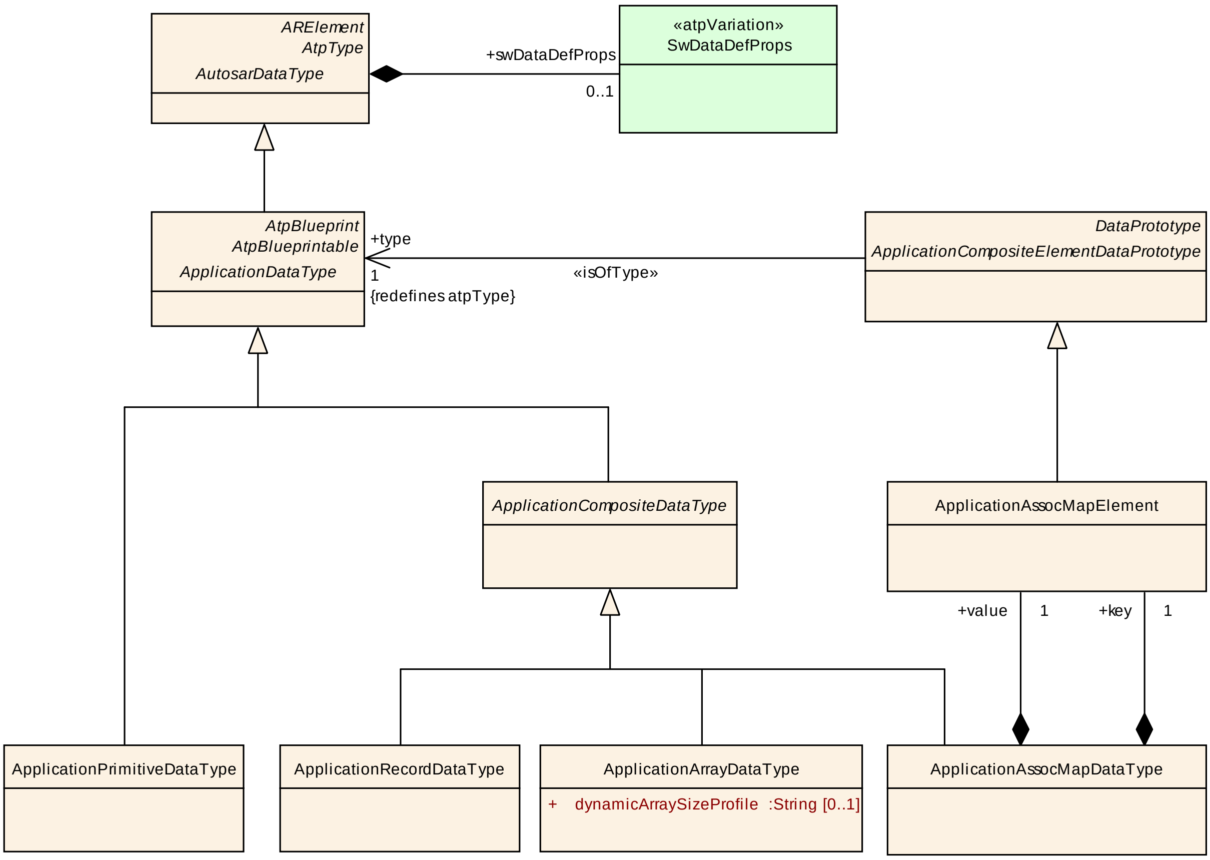

Figure 3.3: Formal model of `ApplicationAssocMapDataType`

Class	**ApplicationAssocMapDataType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DataTypes			
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::DataTypes			
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 ApplicationAssocMap-DataType.

Figure 3.9 contains the corresponding graphical representation of the model.

The corresponding definition of ImplementationDataTypes can be found in Listing 3.4.

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

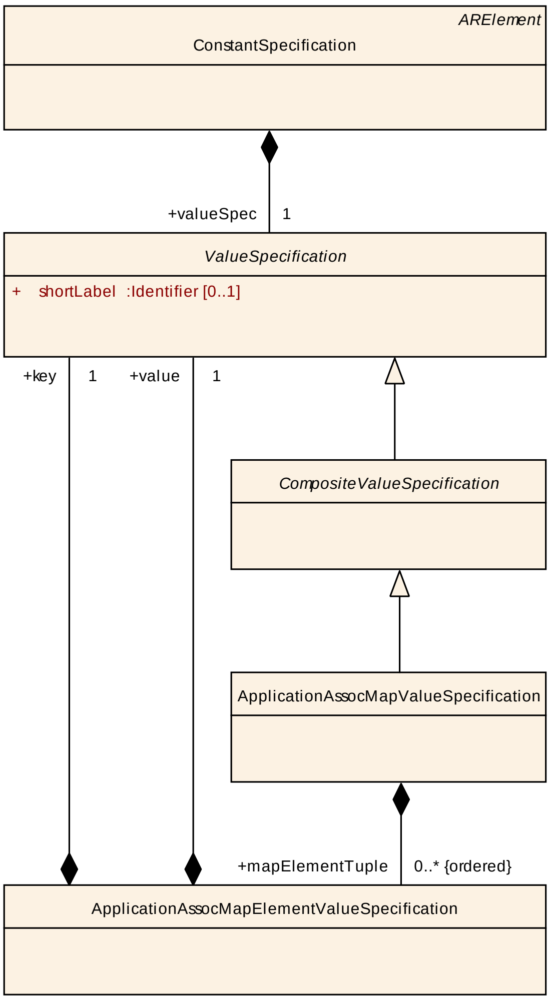

Figure 3.4: Formal model of the initialization of an ApplicationAssocMapDataType

Class	ApplicationAssocMapValueSpecification			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DataTypes			
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::DataTypes			
Note	This meta-class represents the ability to define the initialization of the elements of an ApplicationAssocMapDataType. **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] `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 El- em.		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 ImplementationDataType

[TPS_MANI_01029] Usage of `ImplementationDataType` ⌈ A subset of the modeling of `ImplementationDataType`s that is supported on the *AUTOSAR classic platform* can directly be used on the *AUTOSAR adaptive platform* as well.

In addition to the supported values of `category` on the *AUTOSAR classic platform*, it is possible to use further values that are exclusive to the *AUTOSAR adaptive platform*.
⌋*(RS_MANI_00016)*

[constr_1479] No support for certain values of `ImplementationDataType.category` ⌈ On the *AUTOSAR adaptive platform*, the following values of `ImplementationDataType.category` are not supported:

- `DATA_REFERENCE`
- `FUNCTION_REFERENCE`

⌋*()*

For explanation of the existence of [constr_1479], the utilization of formalized data types on the *AUTOSAR adaptive platform* (currently) extends entirely to communication, there is no description of internal values as it is done extensively on the *AUTOSAR classic platform*.

The usage of pointers (which is what the mentioned two values of `category` represent) is not safe for the purpose of communication that extends potentially beyond the scope of a single process or even machine.

It should be noted that the modeling of variable-size arrays on the *AUTOSAR classic platform* has an intrinsic complexity because the programming language C that is used on the *AUTOSAR classic platform* does not provide a **native** support for variable-size arrays.

The *AUTOSAR adaptive platform*, on the other hand, supports the implementation of software using the programming language C++ [6]. This language comes with built-in so-called *container data-types*.

These container data-types are used to type **objects** (as opposed to a plain piece of data, as used in C), and this fact can be taken to significantly simplify the modeling of existing semantics that is more complex on the *AUTOSAR classic platform*, e.g. the already mentioned variable-size array can be much easier modeled with an underlying C++ `vector`.

Class	**ImplementationDataType**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a ImplementationDataType representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the ImplementationDataType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table 3.11: 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, 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 ImplementationDataTypeElement represents the type of each single array element. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
arraySizeHandling	ArraySizeHandlingEnum	0..1	attr	The way how the size of the array is handled in case of a variable size array.
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls the meaning of the value of the array size.
subElement (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 ImplementationDataType representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElementt.

Table 3.12: ImplementationDataTypeElement

3.3.3.1 String Data Type

The new programming language options for implementing software on the *AUTOSAR adaptive platform* open new ways to define a string data type on the level of `Imple-`

mentationDataType that are less complex than the necessary steps that have to be taken on the *AUTOSAR classic platform*.

For more details about how strings could be used on the *AUTOSAR classic platform*, please refer to the specification of the AUTOSAR Software Component Template [1].

In addition to what is supported on the *AUTOSAR classic platform*, the *AUTOSAR adaptive platform* offers a new value of attribute category of ImplementationDataType: STRING.

[TPS_MANI_01030] ImplementationDataType of category STRING ⌈ An ImplementationDataType of category STRING represents a container data type for a sequence of characters.

AUTOSAR demands that the C++ binding of an ImplementationDataType of category STRING is implemented either by a std::string or by a std::u16string. ⌋*(RS_MANI_00016)*

It is still possible to define an encoding for a string data type according to [TPS_MANI_01030] implemented by a std::string or std::u16string, for any encodings other than ASCII a dedicated library to process the string content would be required.

In other words, these strings should really be thought of as sequences of bytes, where each string type is more suitable for a different Unicode encoding.

[TPS_MANI_03144] C++ language binding of ImplementationDataTypes of category STRING ⌈ The baseTypeSize of the ImplementationDataType that describes the Code Unit size decides about the C++ language binding.

- an ImplementationDataType of category STRING where the baseTypeSize is set to a value of 8 will be implemented as std::string.
- an ImplementationDataType of category STRING where the baseTypeSize is set to a value of 16 will be implemented as std::u16string.

⌋*(RS_MANI_00016)*

[TPS_MANI_03144] means that:

- a String with UTF-8 encoding is always mapped to std::string.
- a String with UTF-16 encoding is always mapped to std::u16string.

[constr_1486] ImplementationDataType of category STRING and SwBaseType ⌈ The ImplementationDataType of category STRING shall aggregate SwDataDefProps in the role swDataDefProps which shall refer to an SwBaseType in the role baseType where the attribute BaseTypeDirectDefinition.baseTypeSize is set to a value of 8 or 16. ⌋*()*

[constr_1475] ImplementationDataType of category STRING is limited ⌈ The usage of an ImplementationDataType of category STRING is limited to the context of AdaptiveApplicationSwComponentTypes and CompositionSwComponentTypes defined in the context of an Executable. ⌋()

[constr_1475] is a formal approach to express that an ImplementationDataType of category STRING shall only be used on the *AUTOSAR adaptive platform*.

The example depicted in Figure 3.5 contains the definition of both an ApplicationDataType as well as the definition of the corresponding ImplementationDataType.

The latter obviously becomes significantly lighter to model thanks to the restriction that, as far as the C++ language binding is concerned, an ImplementationDataType of category STRING shall only be implemented on the basis of a std::string or std::u16string (as expressed by [TPS_MANI_01030]).

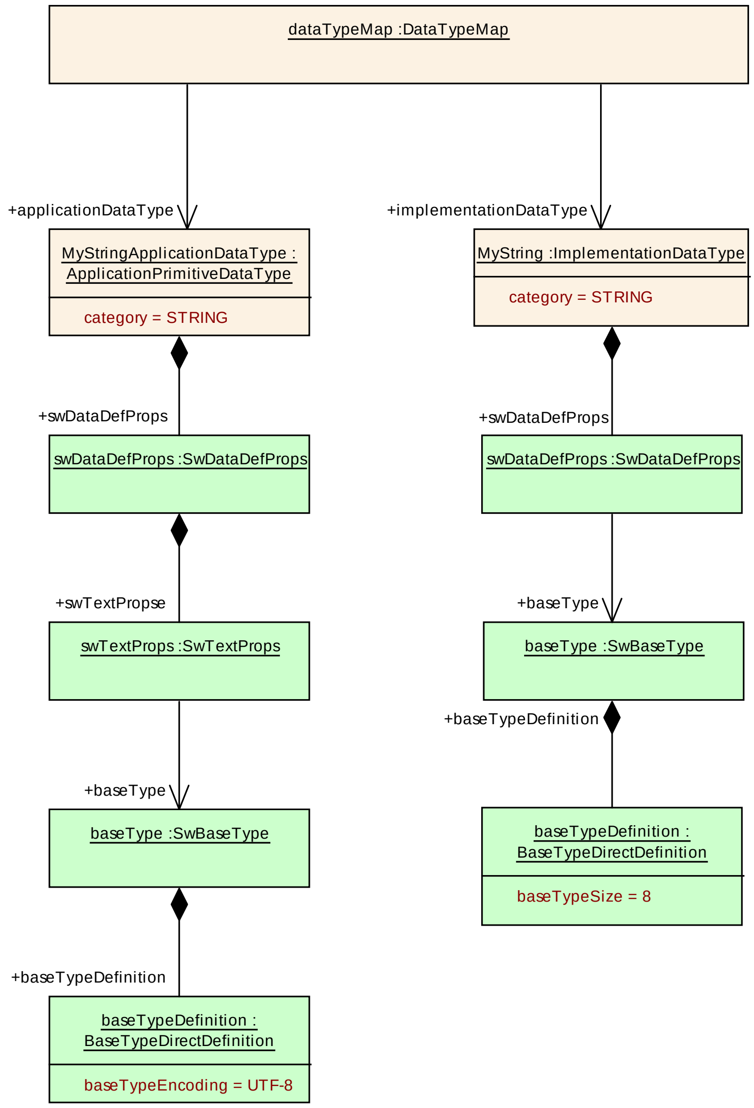

Figure 3.5: Example of the model of a string with UTF-8 encoding

[constr_1485] No `subElement` for `ImplementationDataType` of `category` `STRING` ⌈ `ImplementationDataType` of `category` `STRING` shall not aggregate an `ImplementationDataTypeElement` in the role `subElement`. ⌋*()*

Another aspect of the example in Figure 3.5 is that it defines the intended encoding of the modeled data type in the scope of the `ApplicationPrimitiveDataType`.

This reflects the plausible intention of the creator of the `ApplicationPrimitiveDataType` to take control of the underlying encoding and not leave this decision to the corresponding model of an `ImplementationDataType`.

3.3.3.2 Vector Data Type

There is another case where the language binding to C++ offers new ways of implementing semantics that requires significantly more effort on the *AUTOSAR classic platform*: the so-called variable-size array.

This means that an `ImplementationDataType` of `category` `VECTOR` that holds any data-type other than a further `ImplementationDataType` of `category` `VECTOR` can be taken as the *AUTOSAR adaptive platform* equivalent of an `ImplementationDataType` of `category` `STRUCT` that has attribute `dynamicArraySizeProfile` set to the value `VSA_LINEAR` (see [1]).

On a related note, the companion to an `ApplicationArrayDataType` that does not define attribute `dynamicArraySizeProfile` (which means that the array data type is supposed to have a fixed size) can still be an `ImplementationDataType` of `category` `ARRAY` that is implemented by means of either a `std::array` or a C-style array in C++.

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			
Base	ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, Atp Blueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, Collectable Element, 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	ApplicationArray Element	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 3.13: ApplicationArrayDataType

[TPS_MANI_01018] `ImplementationDataType` of `category` `VECTOR` ⌈ For a C++ binding, an `ImplementationDataType` of `category` `VECTOR` (which can be taken as the equivalent of a variable-size array) can be implemented as a `std::vector` or as a vector type in a custom namespace (e.g. `my::vector`) (provided that the type in the custom namespace can be configured with the available modeling capabilities). ⌋ *(RS_MANI_00016)*

[TPS_MANI_01098] Constraints on the definition of an `ImplementationDataType` of `category` `VECTOR` ⌈ The idea of a container for data that can grow indefinitely has limited compatibility with the requirements of an embedded system, as there are:

- It shall be possible to limit the size of the `ImplementationDataType` of `category` `VECTOR`.
- It shall be possible to control whether the `ImplementationDataType` of `category` `VECTOR` is allocated on the stack or on the heap. This decision has some implications on the safety domain.
- it shall be possible to define the `ImplementationDataType` of `category` `VECTOR` in its own freely defined namespace.

⌋ *(RS_MANI_00016)*

Model elements exist that support the implementation of the topics stated by [TPS_MANI_01098]. The details can be found in Figure 3.6.

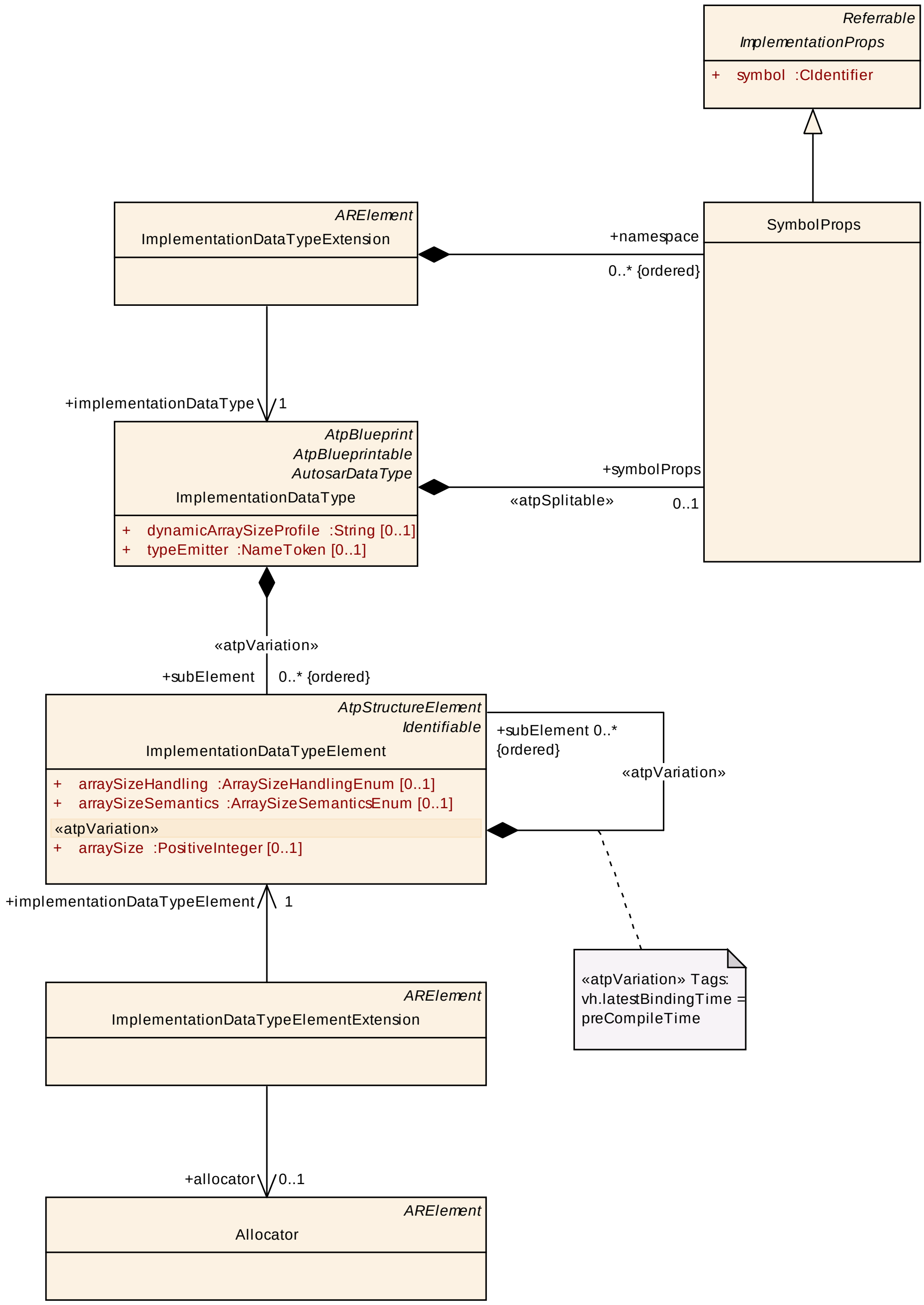

Figure 3.6: C++-specific properties of `ImplementationDataType`

The ability to allocate memory on either stack or heap as well as the assignment of a namespace represent features that are not available on the plain C used on the *AUTOSAR classic platform* for which the `ImplementationDataType` was originally designed.

[TPS_MANI_01099] Semantics of `ImplementationDataTypeElementExtension` ⌈ The meta-class `ImplementationDataTypeElementExtension` has the ability to add semantics to an ImplementationDataType that goes beyond the capabilities of the plain C data type system.

In a true extension behavior, ImplementationDataTypeElementExtension is not a part of the respective ImplementationDataType but references ImplementationDataTypeElement. This way, the extension semantics can be applied without having to touch the definition of the ImplementationDataType ⌋*(RS_MANI_00016)*

Class	**ImplementationDataTypeElementExtension**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DataTypes			
Note	This meta-class represents the ability to define an extension to an ImplementationDataTypeElement to express C++-specific properties. **Tags:** atp.Status=draft; atp.recommendedPackage=ImplementationDataTypeElementExtensions			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
allocator	Allocator	0..1	ref	This represents an allocator taken to create the C++ data type. **Tags:** atp.Status=draft
implementationDataTypeElement	ImplementationDataTypeElement	1	ref	This represents the ImplementationDataTypeElement to extend. **Tags:** atp.Status=draft

Table 3.14: ImplementationDataTypeElementExtension

[constr_1533] Applicability of ImplementationDataTypeElementExtension ⌈ The ability to refer to an ImplementationDataTypeElement in the role ImplementationDataTypeElementExtension.implementationDataTypeElement shall be limited to ImplementationDataType or ImplementationDataTypeElement of category VECTOR. ⌋*()*

For the moment, the existence of [constr_1533] is mainly motivated by the idea to maintain control about the application of ImplementationDataTypeElementExtension.

[TPS_MANI_01100] 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 add the ability to define a maximum number of elements to the semantics of the default allocator of std::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 con-

strained 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.

⌋(*RS_MANI_00016*)

Please note that `Allocator` is derived from `ARElement` in order to make it reusable in different contexts.

[TPS_MANI_01101] Size-constrained allocation of memory ⌈ The size of a memory chunk to be used for a given `ImplementationDataType` or `ImplementationDataTypeElement` of `category` `VECTOR` can be computed out of the information contained in `ImplementationDataTypeElement.arraySize` and the information about the size of one element of the vector. ⌋(*RS_MANI_00016*)

Class	**Allocator**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DataTypes			
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, PackageableElement, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 3.15: Allocator

[TPS_MANI_01102] Specification of a namespace for an `ImplementationDataType` of `category` `VECTOR` ⌈ The ability to define a namespace for a `ImplementationDataType` of `category` `VECTOR` is expressed by means of the aggregation of `SymbolProps` at `ImplementationDataTypeExtension` in the role `namespace`. ⌋(*RS_MANI_00016*)

Class	**ImplementationDataTypeExtension**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::DataTypes			
Note	his meta-class represents the ability to extend the semantics of the ImplementationDataType. **Tags:** atp.Status=draft; atp.recommendedPackage=ImplementationDataTypeExtensions			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
implementationDataType	ImplementationDataType	1	ref	This represents the ImplementationDataType that this subject to the extension. **Tags:** atp.Status=draft

namespace (ordered)	SymbolProps	*	aggr	This represents the intended namespace of the C++ data type **Tags:** atp.Status=draft

Table 3.16: ImplementationDataTypeExtension

[constr_1476] `ImplementationDataType` of `category` `VECTOR` is limited ⌈ The usage of an `ImplementationDataType` of `category` `VECTOR` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s defined in the context of an `Executable`. ⌋*()*

[constr_1476] is a formal approach to express that an `ImplementationDataType` of `category` `VECTOR` shall only be used on the *AUTOSAR adaptive platform*.

An `ImplementationDataType` of `category` `VECTOR` carries the intrinsic semantics that it (bar any limitations set by the used implementation of the C++ runtime) can grow indefinitely.

This technically corresponds to a setting of attribute `dynamicArraySizeProfile` to the value `VSA_FULLY_FLEXIBLE`. In other words, it would not make sense and only lead to confusion if in a concrete model the value of attribute `dynamicArraySizeProfile` would be set to anything else than the value `VSA_FULLY_FLEXIBLE`.

[constr_1506] `ImplementationDataType` of `category` `VECTOR` shall not define `dynamicArraySizeProfile` ⌈ An `ImplementationDataType` of `category` `VECTOR` shall **not define** attribute `dynamicArraySizeProfile`. ⌋*()*

In order to channel the definition of `ImplementationDataType` of `category` `VECTOR` the following rules shall apply:

[TPS_MANI_01042] Definition of a linear `ImplementationDataType` of `category` `VECTOR` ⌈ A **linear** `ImplementationDataType` of `category` `VECTOR` shall aggregate one `ImplementationDataTypeElement` which defines the details of the "payload" of the `ImplementationDataType` of `category` `VECTOR`. ⌋ *(RS_MANI_00016)*

Figure 3.7 contains an example model of a `ImplementationDataType` of `category` `VECTOR`.

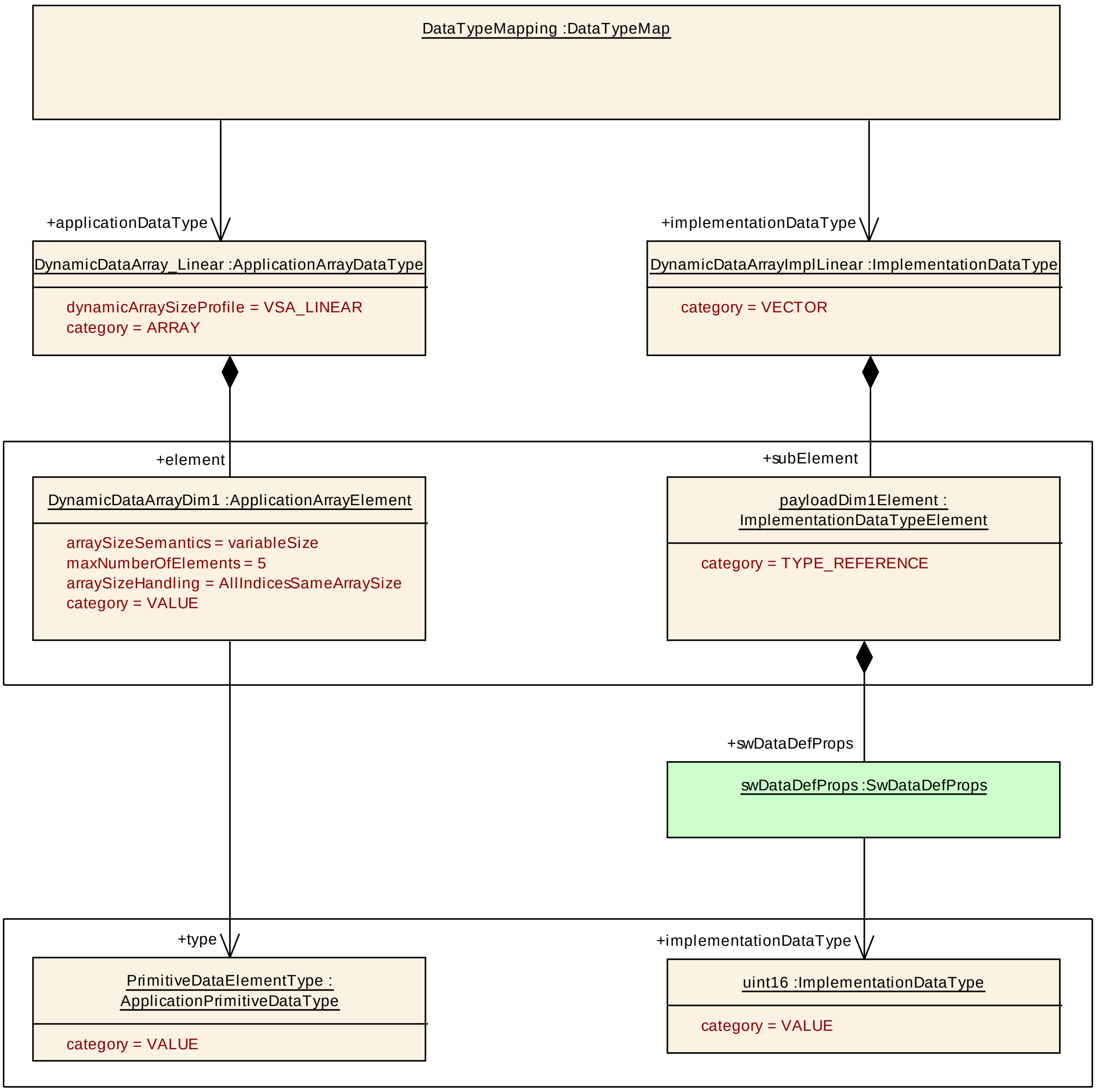

Figure 3.7: A one-dimensional vector

For comparison, the diagram also shows the corresponding ApplicationArrayDataType that has attribute dynamicArraySizeProfile set to the value VSA_LINEAR on the left side.

A corresponding ARXML fragment can be found in Listing 3.2.

Note that the fragment represents only a sketch that concentrates on the most important model elements needed to exemplify the definition of a (semantically) **linear** ImplementationDataType of category VECTOR.

As expected, the usage of an ImplementationDataType of category VECTOR is not limited to one dimension. As a matter of fact, the full range of possible values of attribute dynamicArraySizeProfile (as explained in [TPS_SWCT_01607]) can be used.

Listing 3.2: Example for the definition of a linear ImplementationDataType of category VECTOR

```
<IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>DynamicDataArrayImplLinear</SHORT-NAME>
  <CATEGORY>VECTOR</CATEGORY>
  <SUB-ELEMENTS>
    <IMPLEMENTATION-DATA-TYPE-ELEMENT>
      <SHORT-NAME>payloadDim1Element</SHORT-NAME>
      <CATEGORY>TYPE_REFERENCE</CATEGORY>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">/
               ArrayExamle_VSA_Linear/uint16</IMPLEMENTATION-DATA-TYPE-REF>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
    </IMPLEMENTATION-DATA-TYPE-ELEMENT>
  </SUB-ELEMENTS>
</IMPLEMENTATION-DATA-TYPE>
```

[TPS_MANI_01043] Definition of a rectangular ImplementationDataType of category VECTOR ⌈ A (semantically) **rectangular** ImplementationDataType of category VECTOR shall have the following structure:

- The ImplementationDataType of category VECTOR shall aggregate one ImplementationDataTypeElement where attribute category is set to the value VECTOR.
- The ImplementationDataTypeElement of category VECTOR shall aggregate one further ImplementationDataTypeElement which defines the details of the “payload” of the ImplementationDataType of category VECTOR.

⌋*(RS_MANI_00016)*

Figure 3.8 contains an example model of an ImplementationDataType of category VECTOR that corresponds to [TPS_MANI_01043].

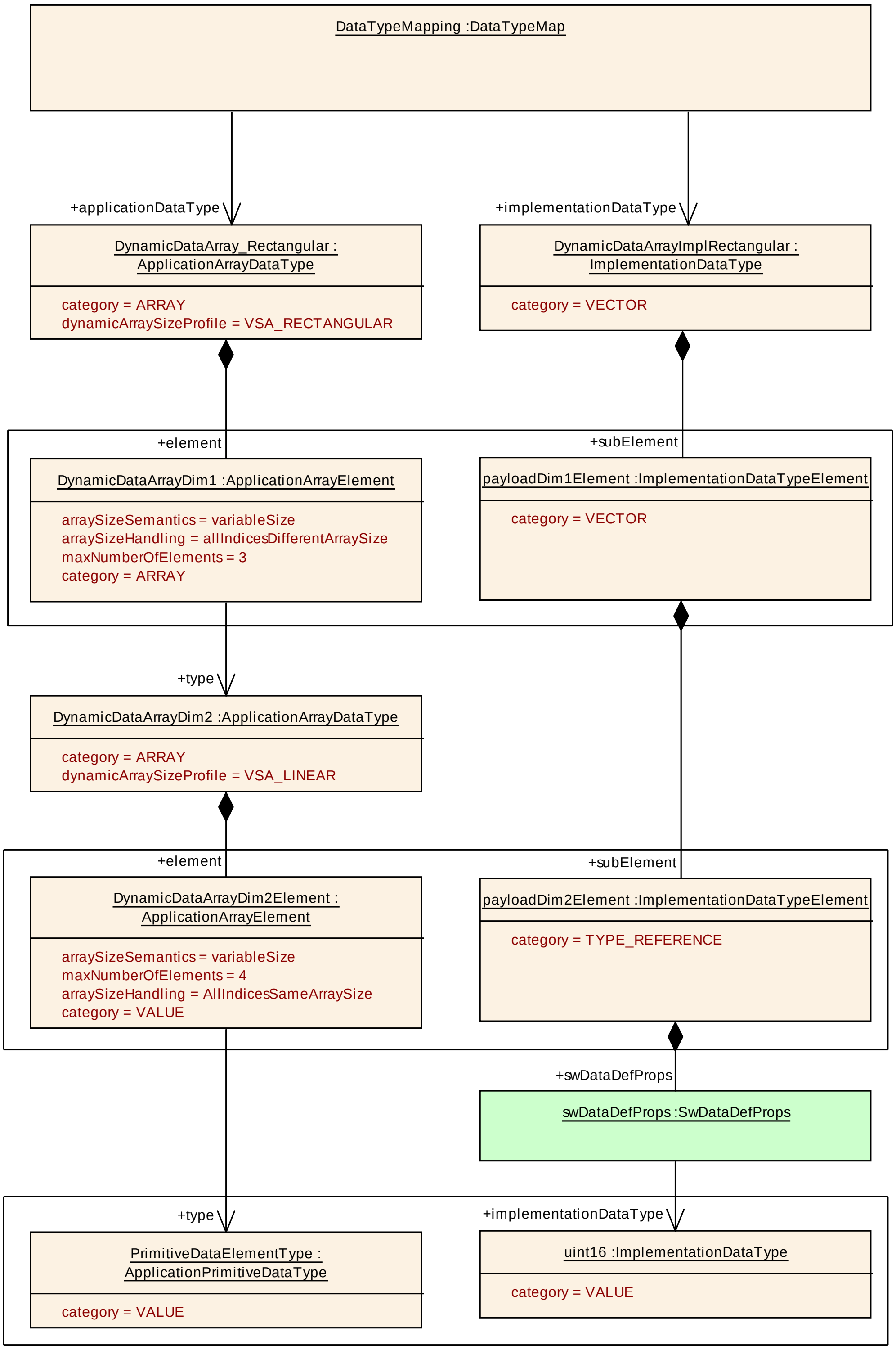

Figure 3.8: A two-dimensional vector with different dimension values

For comparison, the diagram also shows the corresponding `ApplicationArrayDataType` that has attribute `dynamicArraySizeProfile` set to the value `VSA_RECTANGULAR` on the left side.

A corresponding ARXML fragment can be found in Listing 3.3.

Note that the fragment represents only a sketch that concentrates on the most important model elements needed to exemplify the definition of a **rectangular** ImplementationDataType of category VECTOR.

Listing 3.3: Example for the definition of a rectangular ImplementationDataType of category VECTOR

```
<IMPLEMENTATION-DATA-TYPE>
    <SHORT-NAME>DynamicDataArrayImplRectangular</SHORT-NAME>
    <CATEGORY>VECTOR</CATEGORY>
    <SUB-ELEMENTS>
        <IMPLEMENTATION-DATA-TYPE-ELEMENT>
            <SHORT-NAME>payloadDim1Element</SHORT-NAME>
            <CATEGORY>VECTOR</CATEGORY>
            <SUB-ELEMENTS>
                <IMPLEMENTATION-DATA-TYPE-ELEMENT>
                    <SHORT-NAME>payloadDim2Element</SHORT-NAME>
                    <CATEGORY>TYPE_REFERENCE</CATEGORY>
                    <SW-DATA-DEF-PROPS>
                        <SW-DATA-DEF-PROPS-VARIANTS>
                            <SW-DATA-DEF-PROPS-CONDITIONAL>
                                <IMPLEMENTATION-DATA-TYPE-REF DEST="
                                   IMPLEMENTATION-DATA-TYPE">/
                                   ArrayExamle_VSA_Linear/uint16</
                                   IMPLEMENTATION-DATA-TYPE-REF>
                            </SW-DATA-DEF-PROPS-CONDITIONAL>
                        </SW-DATA-DEF-PROPS-VARIANTS>
                    </SW-DATA-DEF-PROPS>
                </IMPLEMENTATION-DATA-TYPE-ELEMENT>
            </SUB-ELEMENTS>
        </IMPLEMENTATION-DATA-TYPE-ELEMENT>
    </SUB-ELEMENTS>
</IMPLEMENTATION-DATA-TYPE>
```

3.3.3.3 Associative Map Data Type

The companion to ApplicationAssocMapDataType on the level of ImplementationDataType could in principle be modeled in various ways.

However, the rules presented in the following paragraphs have been designed to align with an implementation using an std::map on C++[2].

To support this approach a new value of category for ImplementationDataType is necessary.

Since the category value MAP is already taken it is consequently necessary to define a new value that represents the nature of an associative map data type appropriately. This value of category is defined in [TPS_MANI_01028], along with its translation into code.

[2]which is currently the only supported language binding on the *AUTOSAR adaptive platform*

[TPS_MANI_01028] ImplementationDataType of category ASSOCIATIVE_MAP
⌈ An ImplementationDataType of category ASSOCIATIVE_MAP (can be taken as the equivalent of an associative container data structure) shall always be implemented as a std::map for a C++ binding. ⌋*(RS_MANI_00016)*

[constr_1477] ImplementationDataType of category ASSOCIATIVE_MAP is limited ⌈ The usage of an ImplementationDataType of category ASSOCIATIVE_MAP is limited to the context of AdaptiveApplicationSwComponentTypes and CompositionSwComponentTypes defined in the context of an Executable, i.e. such data type shall not be used on the *AUTOSAR classic platform*. ⌋*()*

[constr_1477] is a formal approach to express that an ImplementationDataType of category ASSOCIATIVE_MAP shall only be used on the *AUTOSAR adaptive platform*.

The modeling of an ImplementationDataType of category ASSOCIATIVE_MAP needs to be expressive enough to allow for deriving all necessary information for the language binding.

As a design principle, container data types do not reveal their inner structure to the application programmer, and therefore there is no point in trying to regulate the modeling of such an ImplementationDataType with the goal to mock a std::map as closely as possible.

That said, the conclusion of this observation is that the regulation of the modeling of an ImplementationDataType of category ASSOCIATIVE_MAP can be as simple as possible.

Consequently, [constr_1487] as well as [TPS_MANI_01044] implement this approach.

[constr_1487] Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP ⌈ An ImplementationDataType of category ASSOCIATIVE_MAP shall have exactly two subElements. Their semantic meaning is defined by [TPS_MANI_01044]. ⌋*()*

[TPS_MANI_01044] Structure of an ImplementationDataType of category ASSOCIATIVE_MAP ⌈ An ImplementationDataType of category ASSOCIATIVE_MAP shall have the following structure:

- The **first** ImplementationDataTypeElement aggregated by ImplementationDataType of category ASSOCIATIVE_MAP shall represent the role that corresponds to ApplicationAssocMapDataType.key and define the respective data type details.
- The **second** ImplementationDataTypeElement aggregated by ImplementationDataType of category ASSOCIATIVE_MAP shall represent the role that corresponds to ApplicationAssocMapDataType.value and define the respective data type details.

⌋*(RS_MANI_00016)*

The regulations made by [TPS_MANI_01044] are implemented in the example modeling of an `ImplementationDataType` of `category` `ASSOCIATIVE_MAP` that can be found in Figure 3.9.

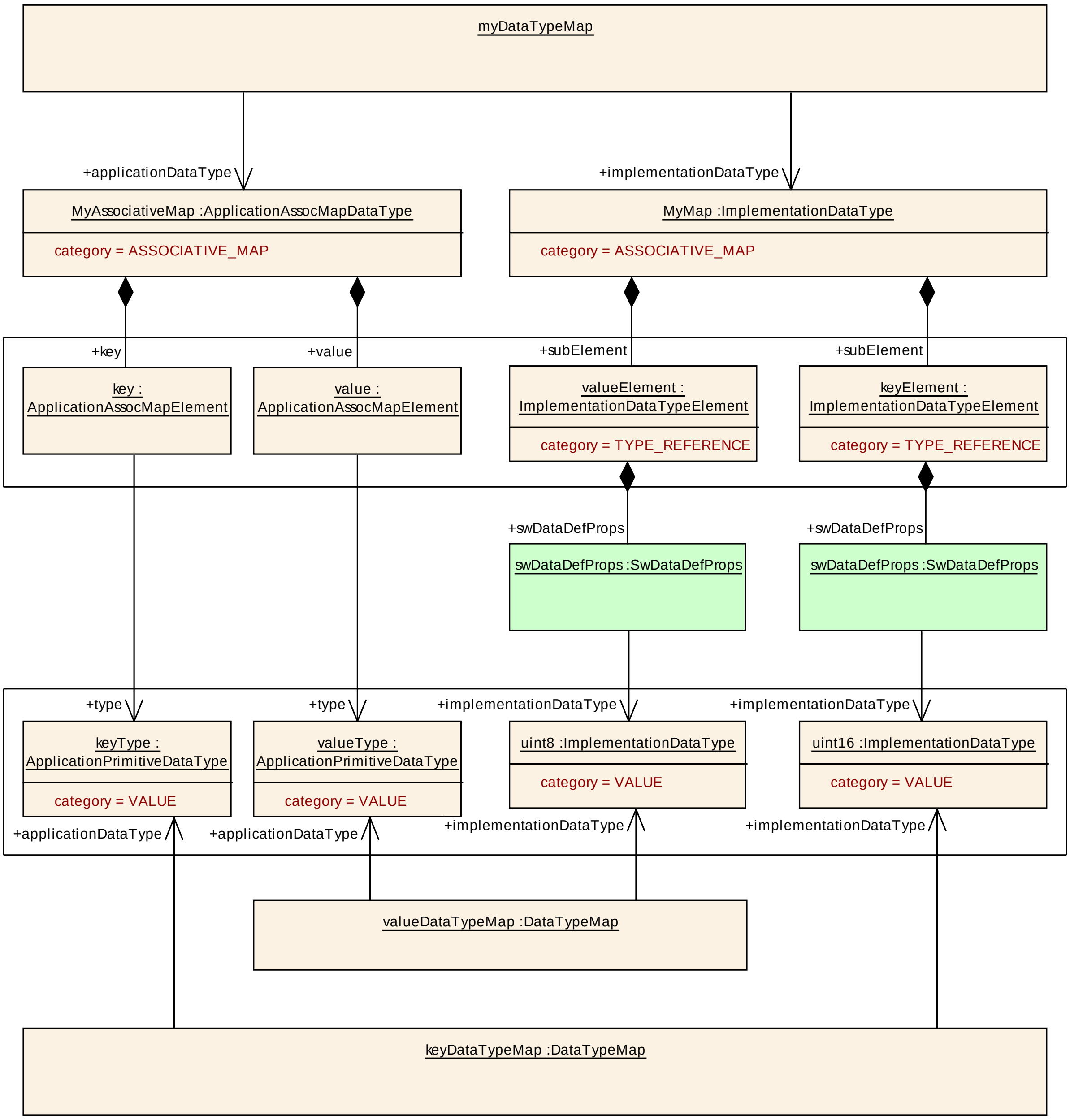

Figure 3.9: Example of the model of an associative map

The ARXML fragment listed in Listing 3.4 corresponds to the model sketched in Figure 3.9. The modeling of the corresponding `ApplicationAssocMapDataType` can be found in Listing 3.1.

Please note the order of definition of `ImplementationDataTypeElement`s in Listing 3.4.

Please note further that the fragments represents only a sketch that concentrates on the most important model elements needed to exemplify the definition of an ImplementationDataType of category ASSOCIATIVE_MAP.

This is significant for the semantics of the overall data type definition, as specified by [TPS_MANI_01044].

Listing 3.4: Example for the definition of an ImplementationDataType of category ASSOCIATIVE_MAP

```
<IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>MyMap</SHORT-NAME>
  <CATEGORY>ASSOCIATIVE_MAP</CATEGORY>
  <SUB-ELEMENTS>
    <IMPLEMENTATION-DATA-TYPE-ELEMENT>
      <SHORT-NAME>keyElement</SHORT-NAME>
      <CATEGORY>TYPE_REFERENCE</CATEGORY>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
               uint16</IMPLEMENTATION-DATA-TYPE-REF>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
    </IMPLEMENTATION-DATA-TYPE-ELEMENT>
    <IMPLEMENTATION-DATA-TYPE-ELEMENT>
      <SHORT-NAME>valueElement</SHORT-NAME>
      <CATEGORY>TYPE_REFERENCE</CATEGORY>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
               uint8</IMPLEMENTATION-DATA-TYPE-REF>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
    </IMPLEMENTATION-DATA-TYPE-ELEMENT>
  </SUB-ELEMENTS>
</IMPLEMENTATION-DATA-TYPE>

<IMPLEMENTATION-DATA-TYPE>
    <SHORT-NAME>uint16</SHORT-NAME>
    <CATEGORY>VALUE</CATEGORY>
</IMPLEMENTATION-DATA-TYPE>

<IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>uint8</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</IMPLEMENTATION-DATA-TYPE>
```

Admittedly, the simplistic approach to modeling an ImplementationDataType of category ASSOCIATIVE_MAP also has its drawbacks.

In a clear departure from the situation on the *AUTOSAR classic platform*, the structure of such an ImplementationDataType does not reflect the structure of a Value-

Specification needed to initialize a corresponding DataPrototype, as already described in section 3.3.2.2.

Finally, the DataTypeMaps depicted in Figure 3.9 can be found in Listing 3.5.

Listing 3.5: Example for the definition of DataTypeMaps for the definition of an associative map data type

```
<DATA-TYPE-MAPPING-SET>
    <SHORT-NAME>MyDataTypeMappingSet</SHORT-NAME>
    <DATA-TYPE-MAPS>
        <DATA-TYPE-MAP>
            <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-ASSOC-MAP-DATA-
               TYPE">MyAssociativeMap</APPLICATION-DATA-TYPE-REF>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
               MyMap</IMPLEMENTATION-DATA-TYPE-REF>
        </DATA-TYPE-MAP>
        <DATA-TYPE-MAP>
            <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-
               TYPE">keyType</APPLICATION-DATA-TYPE-REF>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
               uint16</IMPLEMENTATION-DATA-TYPE-REF>
        </DATA-TYPE-MAP>
        <DATA-TYPE-MAP>
            <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-
               TYPE">valueType</APPLICATION-DATA-TYPE-REF>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
               uint8</IMPLEMENTATION-DATA-TYPE-REF>
        </DATA-TYPE-MAP>
    </DATA-TYPE-MAPS>
</DATA-TYPE-MAPPING-SET>
```

3.3.3.4 Attributes of SwDataDefProps

[constr_1474] SwDataDefProps applicable to ImplementationDataTypes 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 3.17. ⌋*()*

A consequence of [constr_1474] is that the Table 3.17 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		
	ImplementationDataType	ImplementationDataTypeElement	STRING	VECTOR	ASSOCIATIVE_MAP
additionalNativeTypeQualifier					
annotation	x	x	*	*	*
baseType	x	x	1		
compuMethod					
dataConstr	x	x		0..1	
displayFormat	x	x	0..1	0..1	0..1
implementationDataType					
invalidValue	x	x	0..1		
stepSize					
swAddrMethod					
swAlignment					
swBitRepresentation					
swCalibrationAccess					
swCalprmAxisSet					
swComparisonVariable					
swDataDependency					
swHostVariable					
swImplPolicy					
swIntendedResolution					
swInterpolationMethod					
swIsVirtual					
swPointerTargetProps					
swPointerTargetProps.swDataDefProps					
swPointerTargetProps.functionPointerSignature					
swRecordLayout					
swRefreshTiming	x	x	0..1	0..1	0..1
swTextProps					
swValueBlockSize					
unit					
valueAxisDataType					
Other Attributes					
subElement: ImplementationDataTypeElement	x	x		1	2
subElement.arraySizeSemantics	x	x			
subElement.arraySize	x	x			

Table 3.17: Allowed Attributes vs. category for ImplementationDataType

3.3.4 BaseType

Some implications on the usage of data types only occur in the context of the SwBaseType resp. the BaseTypeDirectDefinition.

In other words, there are cases where the data types on the level of ApplicationDataType and ImplementationDataType are identical on both the *AUTOSAR adaptive platform* and the *AUTOSAR classic platform*.

Nevertheless, a different modeling is indicated on the level of the SwBaseType/BaseTypeDirectDefinition (i.e. in the binding to the actual programming language used to implement one of the respective platforms).

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, Collectable Element, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.18: SwBaseType

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	BaseTypeEnco dingString	1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:** xml.sequenceOffset=90
baseType Size	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:** xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:** xml.sequenceOffset=110
maxBaseT ypeSize	PositiveInteger	0..1	attr	Describes the maximum length of the BaseType in bits. **Tags:** xml.sequenceOffset=80

memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:** xml.sequenceOffset=100

nativeDeclaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example
				
				BaseType with
				`shortName: "MyUnsignedInt"`
				`nativeDeclaration: "unsigned short"`
				
				Results in
				`typedef unsigned short MyUnsignedInt;`
				
				If the attribute is not defined the referring ImplementationDataTypes will not be generated as a typedef by RTE.
				
				If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseTypeSize.
				
				This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems.
				
				Tags: xml.sequenceOffset=120

Table 3.19: BaseTypeDirectDefinition

3.3.4.1 Bitfield

A prominent example for this kind of implication is the definition of a `nativeDeclaration` for a primitive type that implements an enumeration or a bitfield.

On the *AUTOSAR classic platform*, support for bitfields and enumeration is possible by using specific kinds of `CompuMethod`s.

However, the language C does not provide portable implementations enumerations or bitfields and thus any bitfields and enumerations can only be implemented by means of plain integer data objects.

This changes on the *AUTOSAR adaptive platform*, here it is possible to use native ways for the implementation of a bitfield, i.e. it is possible to set the value of nativeDeclaration to std::bitset<8> for this purpose.

3.3.4.2 Enumeration

[TPS_MANI_01062] ImplementationDataType to generate a C++ enum ⌈ On the *AUTOSAR adaptive platform*, it is possible to define an ImplementationDataType that refers to a CompuMethod of category TEXTTABLE and use this ImplementationDataType to generate a native C++ enum out of it. ⌋*(RS_MANI_00016)*

[TPS_MANI_01063] Sharing of ImplementationDataType with enumeration semantics ⌈ It is possible to share an ImplementationDataType according to [TPS_MANI_01062] between the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform* if the ImplementationDataType (via SwDataDefProps) does not refer to a SwBaseType where attribute nativeDeclaration exists.

In other words, the ImplementationDataType shall be of category TYPE_REFERENCE and (via SwDataDefProps) refer to another ImplementationDataType that has a shortName that is identical with a shortName of a platform data type. ⌋*(RS_MANI_00016)*

[constr_1508] BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum ⌈ For any given ImplementationDataType, the actual value of the attribute swDataDefProps.baseType.baseTypeDefinition.nativeDeclaration shall **not** be set to the value enum. ⌋*()*

Rationale for the existence of [constr_1508]: the attribute nativeDeclaration is needed for the specification of the integral C++ data type used for the specification of the enumeration.

Note that the usage of attribute SwDataDefProps.additionalNativeTypeQualifier is not required for achieving “native” enum semantics in the generated data type.

On the contrary, the usage of this attribute may potentially complicate the sharing of ImplementationDataTypes between the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*.

Please note further that the definition of an enum is only possible for CompuMethods that represent “pure” enumeration semantics. In cased of a “mixed” semantics (e.g. CompuMethod of category SCALE_LINEAR_AND_TEXTTABLE) it will be necessary to fall back to the generation of symbols in the source code that represent the enumerators.

The details of how an enum data type shall be generated out of the formal definition of an ImplementationDataType are explained in [7].

3.4 Service Interface

3.4.1 Overview

[TPS_MANI_01001] 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 (with
- aggregation of ApplicationError in the role possibleError)

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 ApplicationError 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.

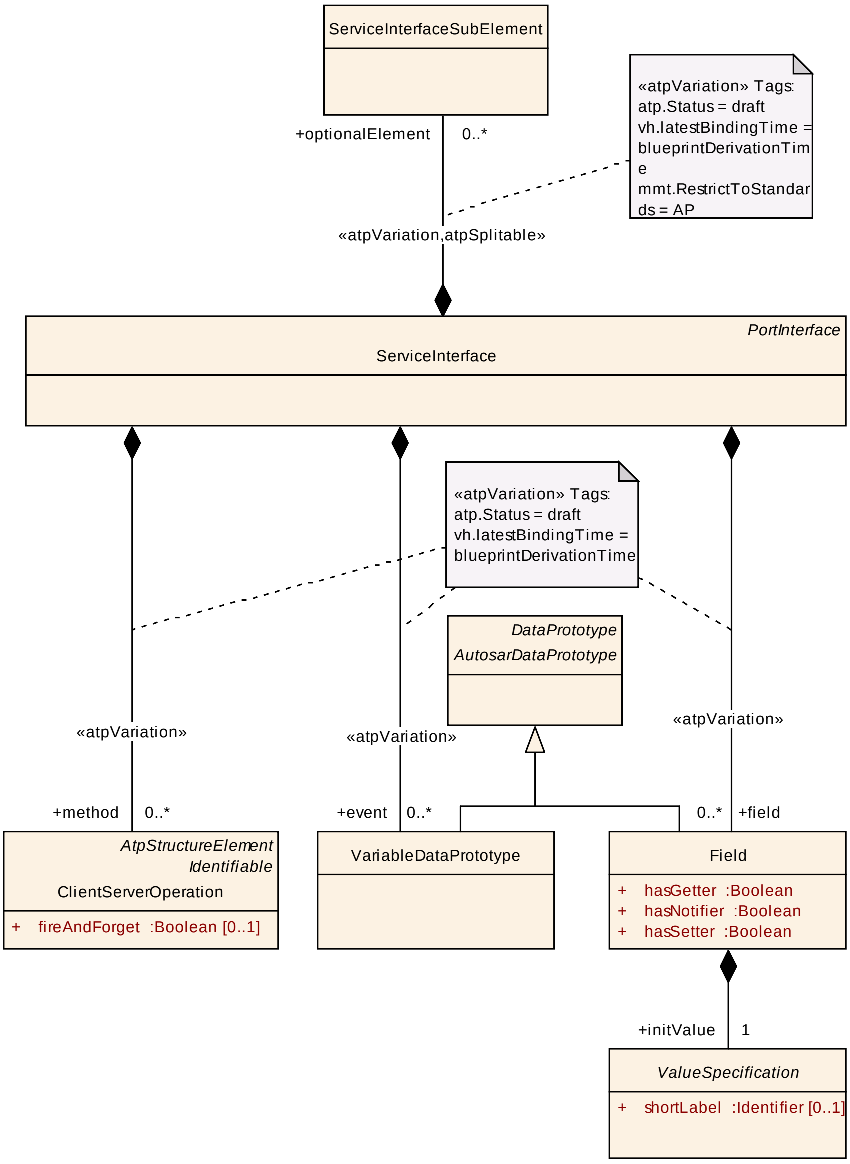

Figure 3.10: Modeling of the ServiceInterface

[constr_1483] 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
optionalElement	ServiceInterfaceSubElement	*	aggr	This aggregation represents the collectionof optional elements within the scope of the enclosing ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel; atp.Status=draft vh.latestBindingTime=blueprintDerivationTime
possibleError	ApplicationError	*	aggr	This represents the collection of ApplicationErrors defined in the context of the enclosing ServiceInterface. **Tags:** atp.Status=draft

Table 3.20: ServiceInterface

[TPS_MANI_01007] 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] 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] 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, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table 3.21: VariableDataPrototype

3.4.3 Field

[TPS_MANI_01034] Semantics of ServiceInterface.field ⌈ A field represents a piece of data hosted by a server that is accessible to one or more client(s) via *get* and/or *set* accessors.

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

[constr_1495] Initial value for field ⌈ A field shall have an initValue. ⌋*()*

If a field defines hasGetter = True then the client may access the value of the Field at any time and at its own discretion. It is therefore necessary that the Field always has a valid value because the client would have no way to distinguish an undefined from a defined value.

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, MultilanguageReferrable, 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.
initValue	ValueSpecification	1	aggr	Specifies initial value(s) of the Field. **Tags:** atp.Status=draft

Table 3.22: Field

3.4.4 Method

[TPS_MANI_01035] 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
possibleError	ApplicationError	*	ref	Possible errors that may by raised by the referring operation.

Table 3.23: 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] 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] 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] 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] method with attribute fireAndForget set to true shall not reference an ApplicationError ⌈ A method that has attribute fireAndForget set to the value true is not allowed to reference an ApplicationError in role possibleError. ⌋*()*

[TPS_MANI_03119] 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 datatypes 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] Compatibility of PortPrototypes of different ServiceInterfaces ⌈ PortPrototypes of different ServiceInterfaces 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] 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] 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] Compatibility of `fields` ⌈ Two `field`s are assumed as compatible if the following conditions apply:

- the two `field`s have identical `shortName`s.
- 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 `AutosarDataType`s of the `AutosarDataPrototype`s. Finally the compatibility rules of the used transport layer will decide whether two `ServiceInterface`s 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 `AutosarDataPrototype`s inside of a `ServiceInterface` are compatible then both `AutosarDataPrototype`s shall be typed by an identical `AutosarDataType`.

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.

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 3.11.

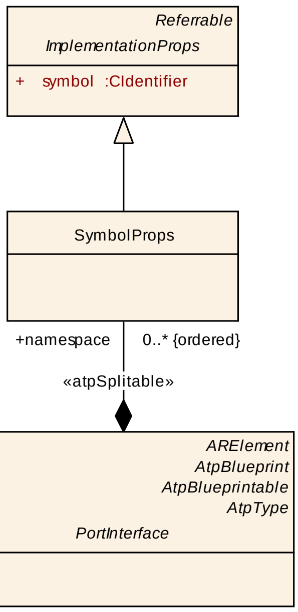

Figure 3.11: Specification of namespaces in PortInterfaces

[TPS_MANI_01004] 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] 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 namespaces has a semantical relevance[3].

[TPS_MANI_01006] 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, followed finally by the shortName of the ServiceInterface. ⌋*(RS_MANI_00003)*

Listing 3.6: 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>
```

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

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			
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 3.24: PortInterface

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

Table 3.25: SymbolProps

The Listing 3.6 exemplifies the statement made by [TPS_MANI_01006], i.e. the resulting name space in e.g. C++ would look like sketched in Listing 3.7.

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

            }
        }
    }
}
```

Listing 3.7: Resulting namespace for the example `ServiceInterface`

3.4.7 Error Handling

[TPS_MANI_01055] Definition of application-level errors ⌈ The `ServiceInterface` aggregates `ApplicationError` in the role `possibleError` in order to allow for the definition of application-level errors. ⌋*(RS_MANI_00003)*

Please note that [constr_1108] specified in AUTOSAR Software Component Template [1] also applies for the possible values of `ApplicationError.errorCode` on the *AUTOSAR adaptive platform*.

[constr_1491] Semantics of `ServiceInterface.possibleError` ⌈ A `ServiceInterface.possibleError` referenced by a given `ClientServerOperation` shall be owned by the same `ServiceInterface` that also owns the `ClientServerOperation`. ⌋*()*

[constr_1522] Semantics of `ClientServerOperation.possibleError` ⌈ An `ApplicationError` referenced by a given `ClientServerOperation` in the role `possibleError` shall only reference `ArgumentDataPrototype`s in the role `errorContext` that are aggregated by the mentioned specific `ClientServerOperation`. ⌋*()*

One problem that the definition of `ApplicationError` by itself doesn't really solve is that the information returned back to the caller in case of an error is extremely limited.

By definition, the **caller cannot rely on the value** of `out`-arguments if an error occurs.

It is, however, considered crucial that the caller has the ability to obtain further information about the nature of an error from the call of a given `ClientServerOperation`. The existence of `ApplicationError.errorContext` fixes this problem.

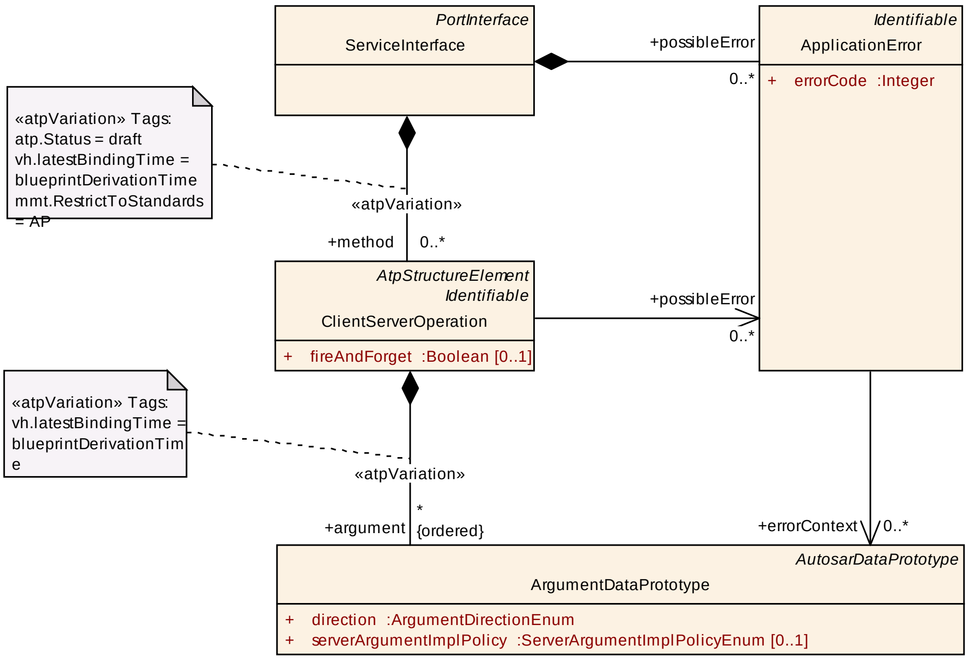

Figure 3.12: Modeling of `ApplicationError` on the *AUTOSAR adaptive platform*

By this means it is **possible to formally identify operation arguments that will have a valid value** if the call to the respective `ClientServerOperation` returns with an error indication.

[TPS_MANI_01056] Semantics of ApplicationError.errorContext ⌈ ArgumentDataPrototypes referenced in the role ApplicationError.errorContext are used to convey context information about a given error scenario back to the caller.

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).
errorContext	ArgumentDataPrototype	*	ref	This reference identifies out arguments that shall have a meaning only if an error occurs. **Tags:** atp.Status=draft; atp.StatusComment=Reserved for AUTOSAR adaptive platform

Table 3.26: ApplicationError

Therefore, if an error occurs then ArgumentDataPrototypes referenced in the role ApplicationError.errorContext shall (in contrast to ArgumentDataPrototypes **not** referenced in this role) have a valid value upon termination of the execution of the associated ClientServerOperation. ⌋()

[constr_1493] ArgumentDataPrototype referenced in the role ApplicationError.errorContext ⌈ The reference to ArgumentDataPrototype in the role ApplicationError.errorContext is **only** supported for ArgumentDataPrototypes where attribute direction is set to out. ⌋()

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 and to the value useArrayBaseType for arrays.

Table 3.27: ArgumentDataPrototype

Enumeration	**ArgumentDirectionEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types
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
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 3.28: ArgumentDirectionEnum

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 `ApplicationDataType`s for the specification of the details of the `ServiceInterface`.

It is therefore necessary to provide the definition of an `ImplementationDataType` for each of the used `ApplicationDataType`. In the meta-model, this correspondence is implemented by means of the meta-class `DataTypeMappingSet`[4].

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

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 ImplementationDataType 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.

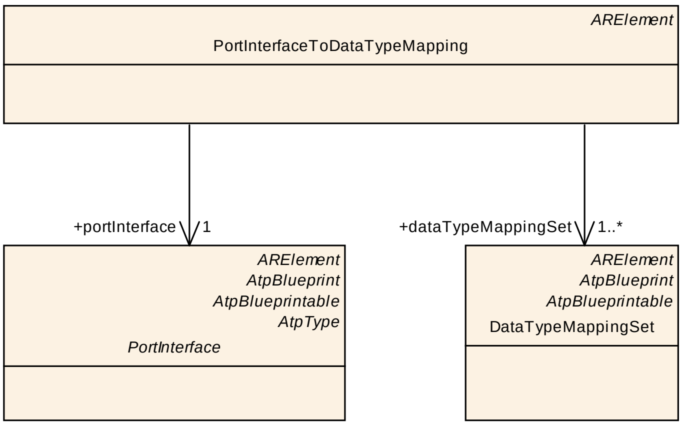

Figure 3.13: Modeling of PortInterfaceToDataTypeMapping

[constr_1507] PortInterfaceToDataTypeMapping is only applicable to ServiceInterface ⌈ PortInterfaceToDataTypeMapping.portInterface shall only refer to a ServiceInterface. ⌋*()*

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			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataTypeMappingSet	DataTypeMappingSet	1..*	ref	This represents the reference to the applicable dataTypemappingSet **Tags:** atp.Status=draft; atp.StatusComment=Reserved for adaptive platform
portInterface	PortInterface	1	ref	This represents the reference to the applicable PortInterface **Tags:** atp.Status=draft; atp.StatusComment=Reserved for adaptive platform

Table 3.29: 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			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an ApplicationDataType and its ImplementationDataType.
modeRequestTypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an ModeDeclarationGroup and its ImplementationDataType.

Table 3.30: DataTypeMappingSet

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

Table 3.31: 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.

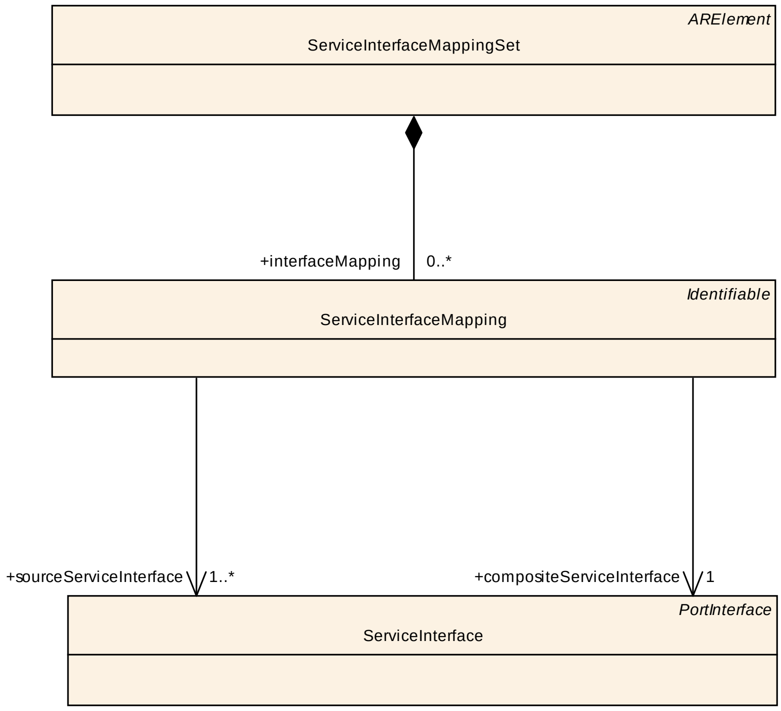

Figure 3.14: Modeling of the ServiceInterfaceMapping

[TPS_MANI_01002] 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] 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 Application Manifest.

⌋(*RS_MANI_00017*)

Figure 3.15 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.

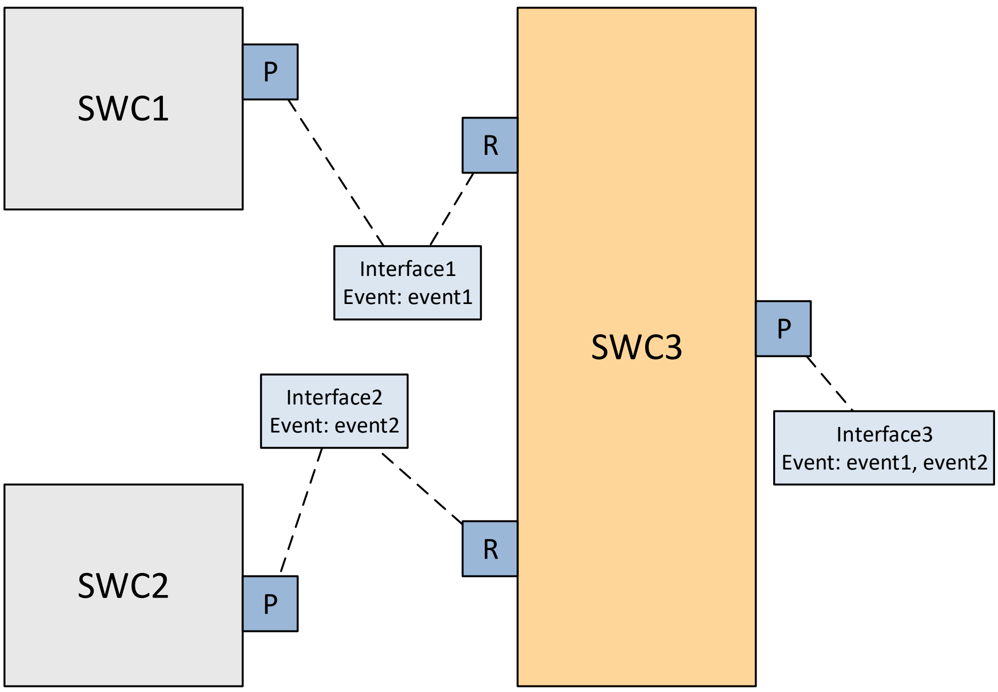

Figure 3.15: Concept of a facade software-component

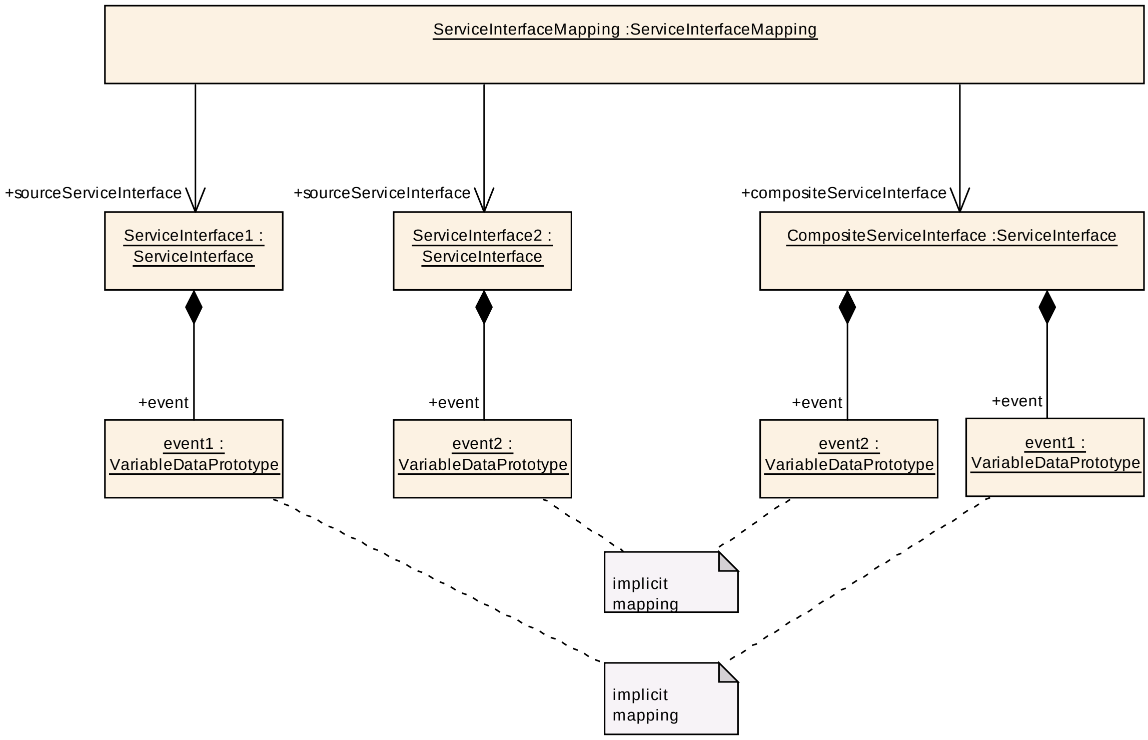

Figure 3.16: Example for the application of a ServiceInterfaceMapping

[TPS_MANI_01022] 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 3.16.

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::ServiceInterface Mapping			
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 3.32: ServiceInterfaceMapping

Class	ServiceInterfaceMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
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 3.33: ServiceInterfaceMappingSet

[TPS_MANI_01003] Limitation of the applicability of `ServiceInterfaceMapping` ⌈The applicability of the `ServiceInterfaceMapping` is limited to cases where the `shortName`s 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 `ServiceInterfaceMapping`s as well as the `ServiceInterfaceElementMapping` introduced in section 3.6.

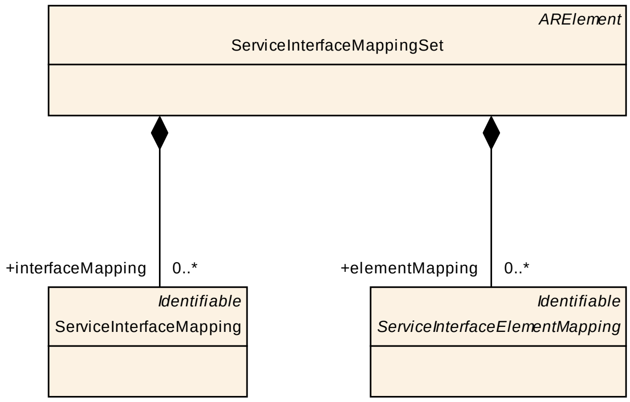

Figure 3.17: Modeling of the ServiceInterfaceMappingSet

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 3.18 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

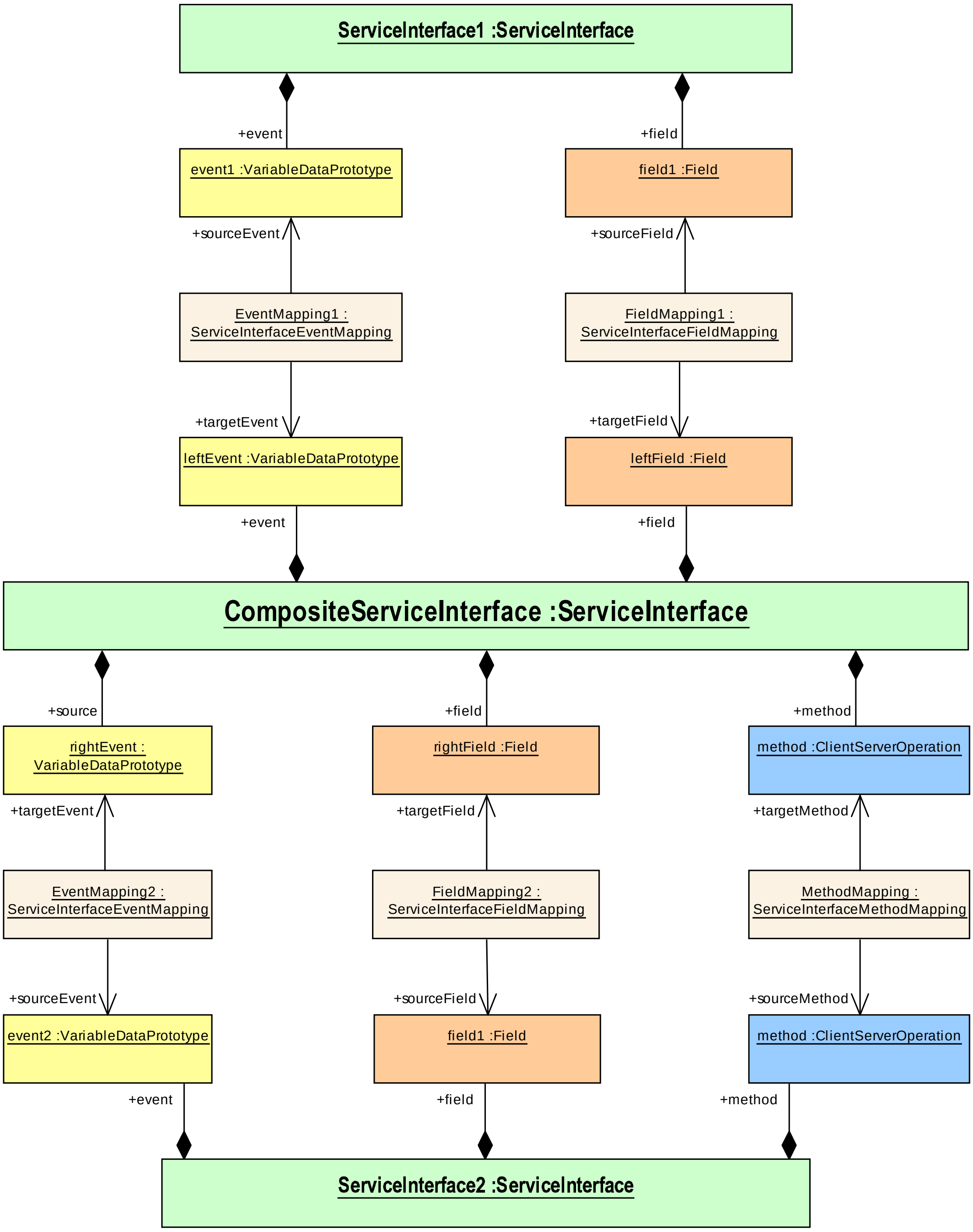

Figure 3.18: Example for a mapping of elements of `ServiceInterface`

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] 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::ServiceInterface Mapping			
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			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 3.34: 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] 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)*

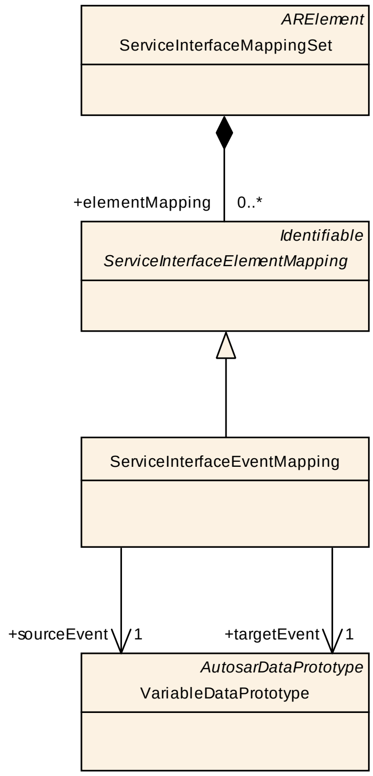

Figure 3.19: Modeling of the ServiceInterfaceEventMapping

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 3.35: ServiceInterfaceEventMapping

The explicit mapping implemented by `ServiceInterfaceEventMapping` does **not** require equal `shortName`s 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 3.20.

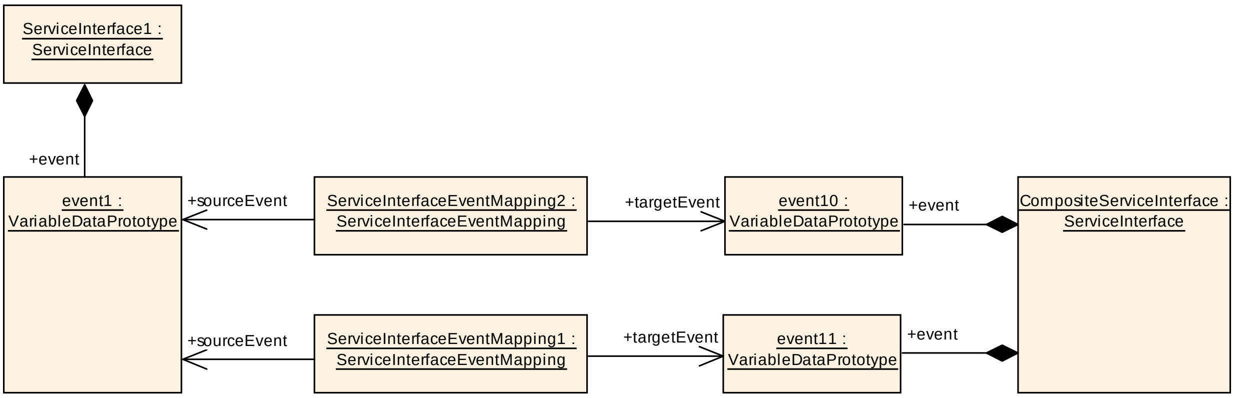

Figure 3.20: Example for the application of a ServiceInterfaceEventMapping

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] 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)*

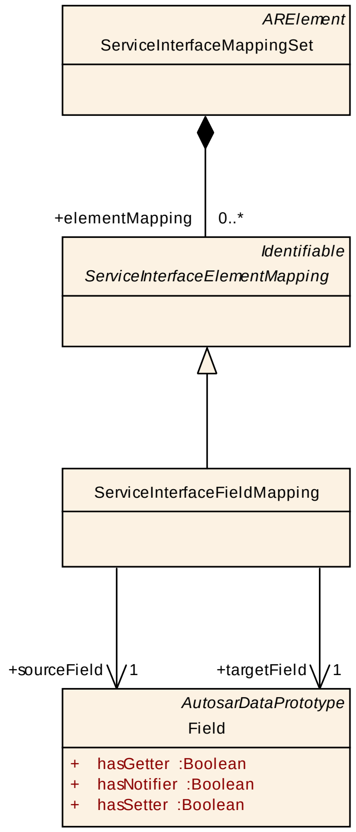

Figure 3.21: Modeling of the ServiceInterfaceFieldMapping

Class	**ServiceInterfaceFieldMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
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, ServiceInterfaceElement Mapping			
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 3.36: 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 3.22.

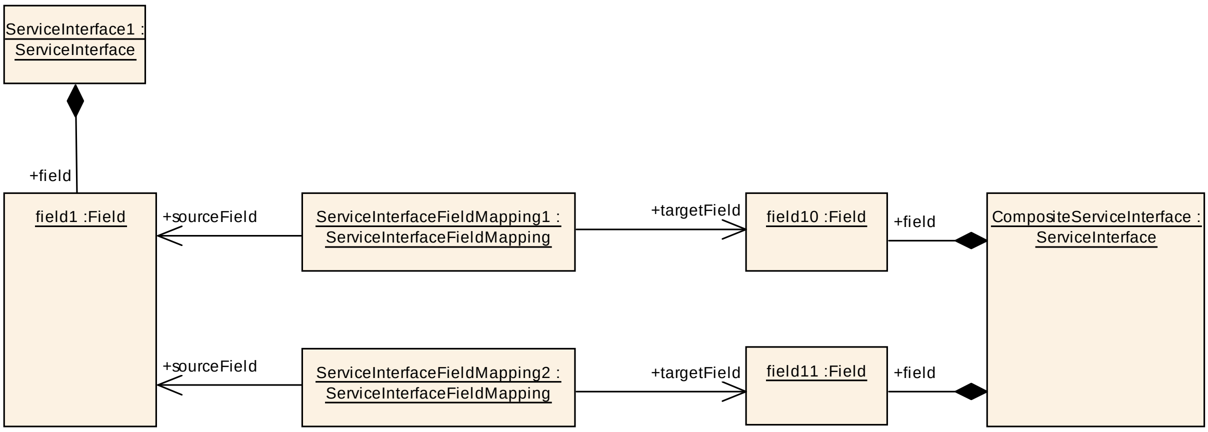

Figure 3.22: Example for the application of a ServiceInterfaceFieldMapping

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] 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)*

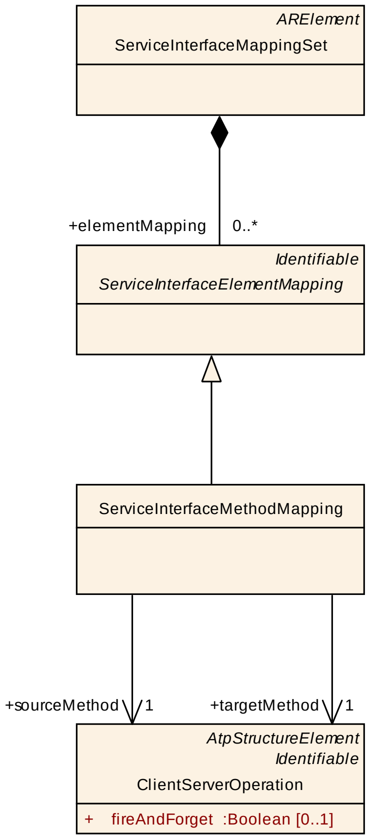

Figure 3.23: Modeling of the `ServiceInterfaceMethodMapping`

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 3.37: ServiceInterfaceMethodMapping

The explicit mapping implemented by `ServiceInterfaceMethodMapping` does **not** require equal `shortName`s 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 3.24.

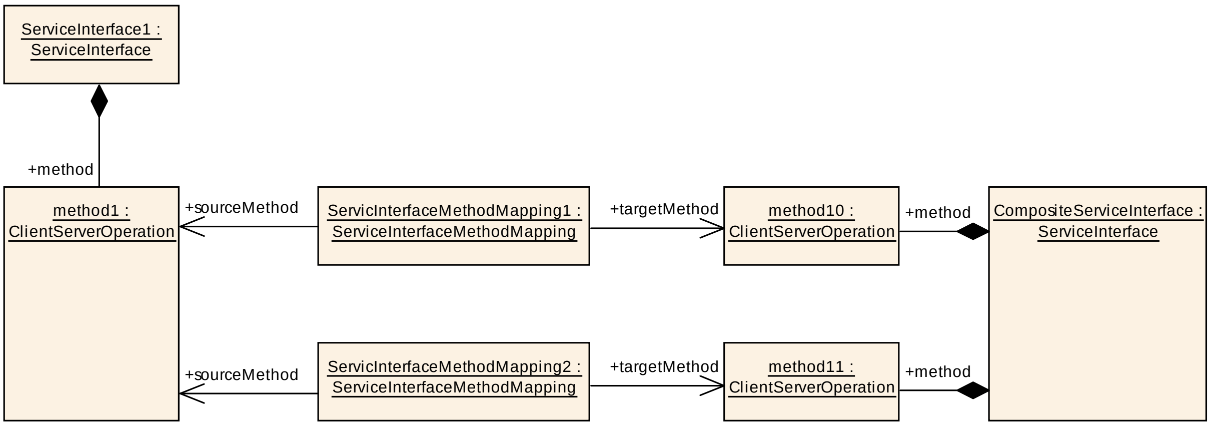

Figure 3.24: Example for the application of a ServiceInterfaceMethodMapping

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.

3.6.5 Service Interface Application Error Mapping

[TPS_MANI_01058] Ability to create a mapping of ApplicationErrors aggregated in the role possibleError ⌈ Apart from the “first-class citizen” of a ServiceInterface, i.e. event, method, and field, there is also the ability to create a mapping of ApplicationErrors aggregated in the role possibleError. ⌋ *(RS_MANI_00017)*

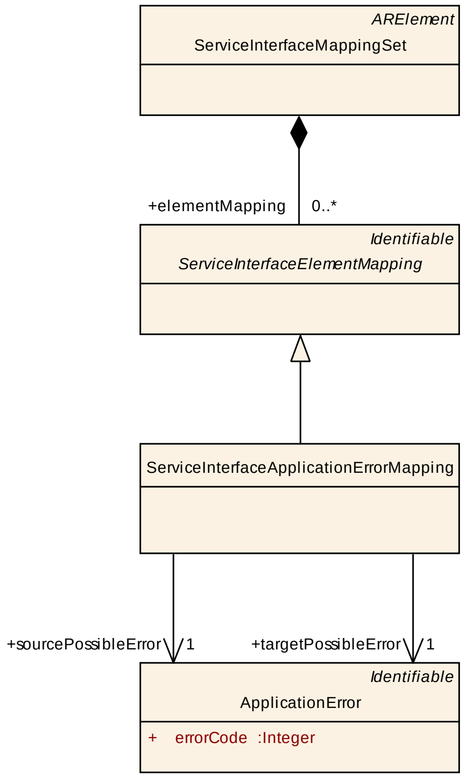

Figure 3.25: Modeling of the `ServiceInterfaceApplicationErrorMapping`

Class	**ServiceInterfaceApplicationErrorMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
Note	This meta-class allows to define a mapping between possibleErrors of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceInterfaceElement Mapping			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
sourcePos sibleError	ApplicationError	1	ref	This reference represents the source end of the ApplicationError mapping. **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform
targetPossi bleError	ApplicationError	1	ref	This reference represents the target end of the ApplicationError mapping **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform

Table 3.38: ServiceInterfaceApplicationErrorMapping

3.7 Persistency Interface

3.7.1 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).

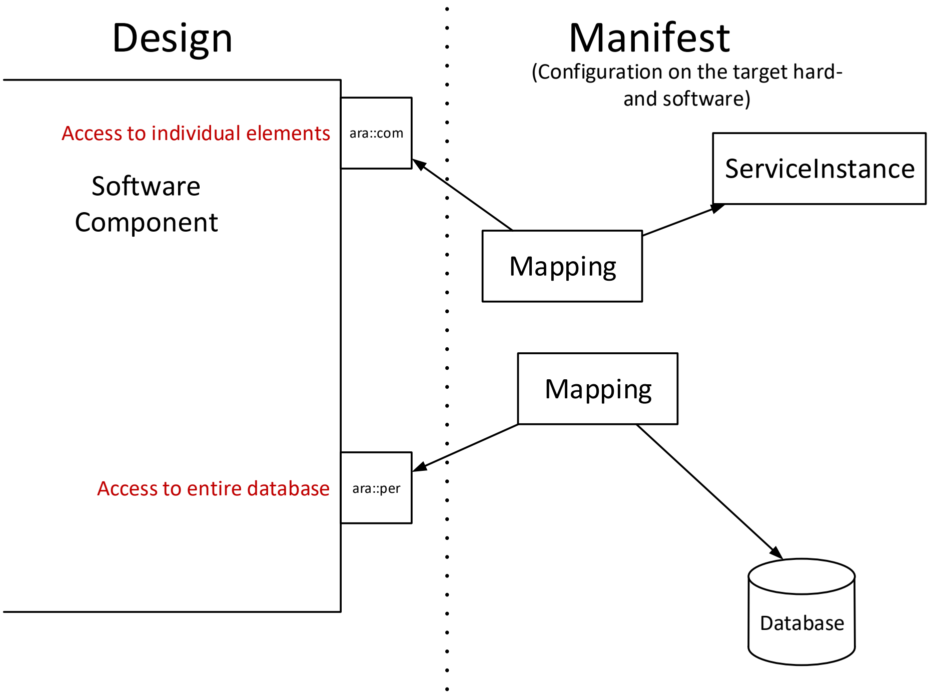

Figure 3.26: General approach for the modeling of persistency

The aspect of deployment for the configuration of persistent data is explained in Figure 3.26.

Please note that the AUTOSAR meta-model actually defines two separate meta-classes (for more details, please refer to Figure 3.27) for the different use cases of access to persistent data (i.e. PersistencyKeyValueDatabaseInterface) and access to files on the file system (by means of PersistencyFileProxyInterface).

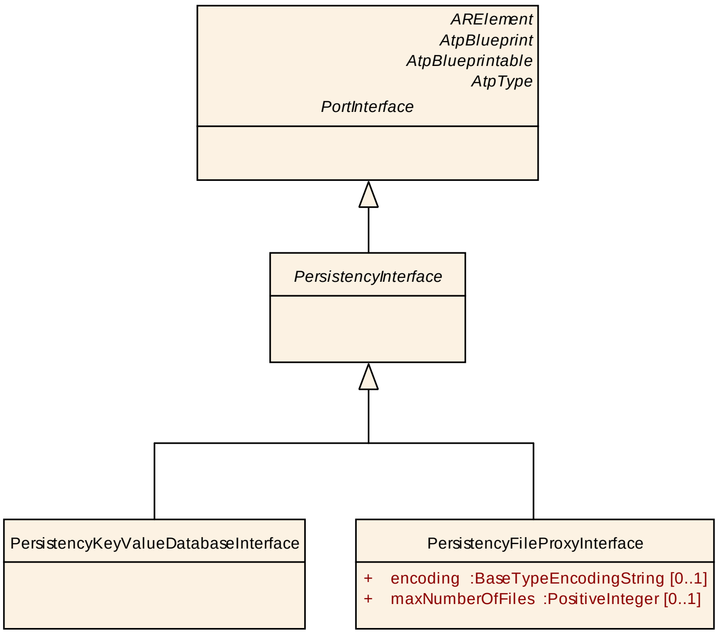

Figure 3.27: Specification of `PortInterface`s for persistency use cases

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, Port Interface, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.39: PersistencyInterface

3.7.2 Persistency Key Value Database Interface

[TPS_MANI_01065] 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=PersistencyKeyValueDatabase Interfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PersistencyInterface, PortInterface, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.40: PersistencyKeyValueDatabaseInterface

3.7.3 Persistency File Proxy Interface

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

[constr_1524] 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] Semantics of `PersistencyFileProxyInterface.maxNumberOfFiles` ⌈ Any `PortPrototype` typed by a `PersistencyFileProxyInterface` implements arrays semantics with respect to the file content. In other words, the `PortPrototype` represents a number of files as opposed to just a single file.

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`. ⌋*(RS_MANI_00027)*

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).
maxNumberOfFiles	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 3.41: PersistencyFileProxyInterface

Please note that the existence of the `PersistencyFileProxyInterface` does not violate the restrictions set by the POSIX subset PSE51 defined in IEEE1003.13 [8].

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*.

3.8 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.

3.8.1 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] No support for PRPortPrototype ⌈ A ServiceInterface shall not be referenced by a PRPortPrototype in the role providedRequiredInterface. ⌋*()*

[TPS_MANI_01039] 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] 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 3.8.6.
- The usage of elements of the corresponding ServiceInterface may need to be configured for a given PortPrototype. This aspect is discussed in section 3.8.7.

3.8.2 Interaction with Crypto Software

[TPS_MANI_01087] Interaction with crypto software ⌈ Interaction with crypto software on an instance of the *AUTOSAR adaptive application* shall be modeled on the basis of the existence of PortPrototypes typed by ClientServerInterfaces ⌋*(RS_MANI_00031)*

The specific shape of the ClientServerInterfaces used for this purpose is defined in the SWS AdapiveCryptoInterface [9].

[TPS_MANI_01088] Semantics of CryptoNeed ⌈ The meta-class CryptoNeed allows for the specification of the general strategy (e.g. the nature of the crypto algorithm) for the application of a crypto API. ⌋*(RS_MANI_00031)*

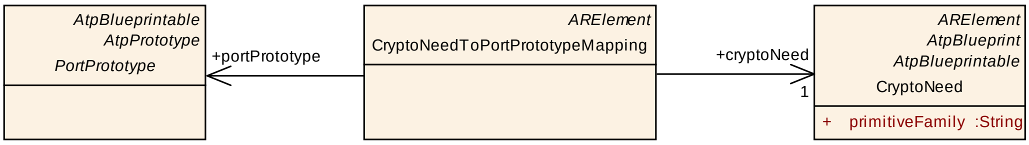

Figure 3.28: Modeling of the relation between CryptoNeed and PortPrototype

[TPS_MANI_01089] Relation between CryptoNeed and PortPrototype ⌈ The meta-class CryptoNeedToPortPrototypeMapping can be taken to describe a concrete relation between a given CryptoNeed and a PortPrototype typed by a ClientServerInterface that is supposed to provide a crypto API. ⌋ *(RS_MANI_00031)*

Please note that the semantics of CryptoNeed in principle could have been modeled by means of a SwcServiceDependency and an aggregated ServiceNeeds.

However, this requires the definition of a software-component. On the other hand, the definition of a CryptoNeed is typically created by an OEM in an early stage of a development project.

Although the individual development approach may vary by OEM, it could easily happen that the design work done by the OEM does not extend to the definition of software-components.

In other words, an OEM that chooses to let the software-component structure be designed by suppliers would not be able to contribute the semantics that is contained in the CryptoNeeds.

[constr_1529] Standardized values of CryptoNeed.category ⌈ The following values of CryptoNeed.category are standardized by AUTOSAR:

- PAYMENT_INFORMATION
- PERSONAL_IDENTIFIABLE_INFORMATION

⌋*()*

Beyond the regulation made by [constr_1529] it is possible to assign custom values of CryptoNeed.category.

In this case, however, it is mandatory to use a company-specific prefix or suffix to the custom values in order to positively avoid clashes with potential future extensions of the collection of standardized values defined by [constr_1529].

Class	**CryptoNeed**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents a statement regarding the applicable crypto use case. **Tags:** atp.Status=draft; atp.recommendedPackage=CryptoNeeds			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
primitiveFamily	String	1	attr	This attribute represents the ability to specify the algorithm family of the crypto need. **Tags:** atp.Status=draft

Table 3.42: CryptoNeed

Class	**CryptoNeedToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to map a crypto need onto a PortPrototype. **Tags:** atp.Status=draft; atp.recommendedPackage=CryptoNeedToPortPrototypeMappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
cryptoNeed	CryptoNeed	1	ref	This meta-class represents the crypto need part of the mapping from crypto need to PortPrototype. **Tags:** atp.Status=draft
portPrototype	PortPrototype	1	ref	This meta-class represents the PortPrototype part of the mapping from crypto need to PortPrototype. **Tags:** atp.Status=draft

Table 3.43: CryptoNeedToPortPrototypeMapping

3.8.3 Interaction with Persistent Storage

The usage of `PortPrototype`s for the purpose of interacting with persistent 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] 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)*

It is possible to model whether and how the files that correspond to the PortPrototype shall be encrypted from the perspective of the designer of the software-component. Details of the approach are further explained in section 3.8.2 as well as Figure 3.28.

[TPS_MANI_01077] Specification of file encryption ⌈ Cryptographic methods can be applied to a PortPrototype typed by a PersistencyFileProxyInterface by referencing the PortPrototype from a CryptoNeedToPortPrototypeMapping that also refers to a CryptoNeed that provides further details about the nature of the applicable cryptographic algorithms. ⌋*(RS_MANI_00027)*

3.8.4 Interaction with Files

Interaction with files can involve the ability to read from and write to a files by the same application. Therefore, the existence of a PRPortPrototype typed by a PersistencyFileProxyInterface shall be supported.

[TPS_MANI_01081] 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 resp. 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 resp. created for read and write access**. Also, there is the ability to **delete** a file.

⌋*(RS_MANI_00027)*

3.8.5 Interaction with Platform Health Management

Platform health management functional cluster within the Adaptive Platform is responsible to supervise the execution of applications, monitor their status, provide rule-based evaluation and execution 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 SwcServiceDependency and ServiceNeeds which can be applied to many use-cases concerning the interaction of application software with platform software.

Please note that because of the re-use of the meta-classes of the AUTOSAR classic platform the terms *Watchdog Manager* and the abbreviation *WdgM* still occur in the descriptions. These terms may be used synonymous for *platform health management* and will be clean up later.

3.8.5.1 Interaction with supervised entities and checkpoints

The interaction of supervision with the platform health management is defined by two interacting structures: checkpoints and supervised entities.

[TPS_MANI_03500] Definition of platform health management checkpoints ⌈ The meta-class SwcServiceDependency together with a SupervisedEntityCheckpointNeeds and a RoleBasedPortAssignment is used to define the interaction of one checkpoint with the platform health management supervision (see figure 3.29). ⌋ *(RS_MANI_00032)*

The referenced required PortPrototype from the RoleBasedPortAssignment represents one checkpoint. The PortPrototype is typed by a PortInterface which details are defined by the platform health management specification.

The application code then calls the *CheckpointReached* method of the respective PortPrototype in order to notify the platform health management that this specific checkpoint has been reached in the program flow.

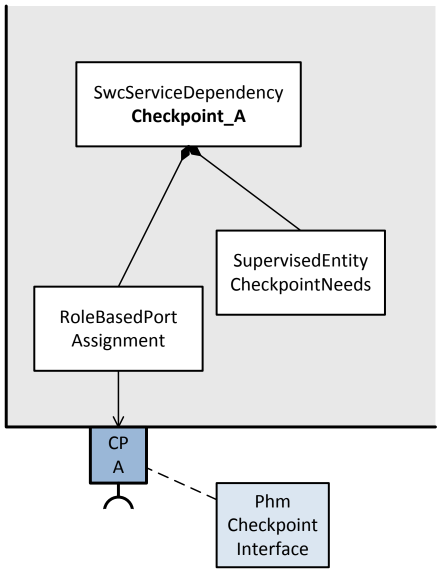

Figure 3.29: SwcServiceDependency and SupervisedEntityCheckpointNeeds

One checkpoint represented by a PortPrototype and the respective SwcServiceDependency is one part of the definition of supervision for the platform health management. The second part is the organization of checkpoints into a supervised entities.

[TPS_MANI_03501] Definition of platform health management supervised entities ⌈ The meta-class SwcServiceDependency referencing the included SupervisedEntityCheckpointNeeds together with a SupervisedEntityNeeds and a RoleBasedPortAssignment is used to define the interaction of one supervised entity with the platform health management supervision (see figure 3.30). ⌋ *(RS_MANI_00032)*

If the application wants to query the status of a supervised entity monitored by the platform health management the the application code calls the *GetSupervisionStatus* method of the respective PortPrototype.

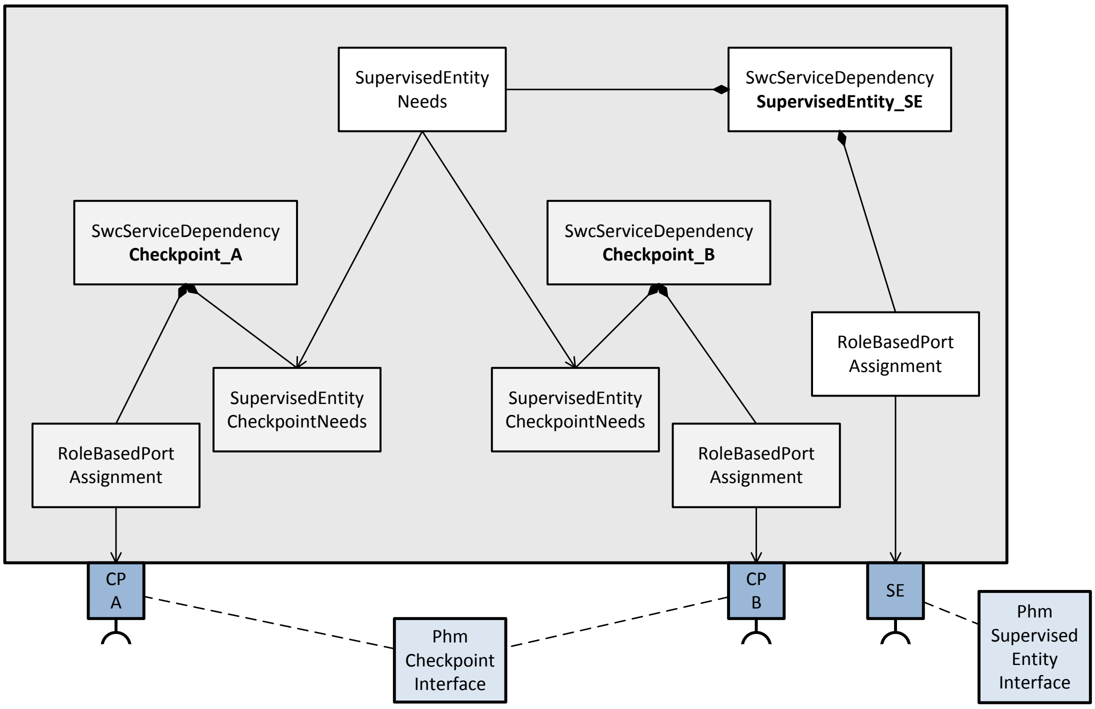

Figure 3.30: SwcServiceDependency and SupervisedEntityNeeds

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 14.2.

Class	**SupervisedEntityNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Watchdog Manager for one specific Supervised Entity.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
activateAtStart	Boolean	1	attr	True/false: supervision activation status of SupervisedEntity shall be enabled/disabled at start.
checkpoints	SupervisedEntityCheckpointNeeds	*	ref	This reference indicates the checkpoints belonging to the Supervised Entity. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
enableDeactivation	Boolean	1	attr	True: software-component shall be allowed to deactivate supervision of this SupervisedEntity false: software-component shall be not allowed to deactivate supervision of this SupervisedEntity

expectedAliveCycle	TimeValue	1	attr	Expected cycle time of alive trigger of this SupervisedEntity (in seconds).
maxAliveCycle	TimeValue	1	attr	Maximum cycle time of alive trigger of this SupervisedEntity (in seconds).
minAliveCycle	TimeValue	1	attr	Minimum cycle time of alive trigger of this SupervisedEntity (in seconds).
toleratedFailedCycles	PositiveInteger	1	attr	Number of consecutive failed alive cycles for this SupervisedEntity which shall be tolerated until the supervision status of the SupervisedEntity is set to WDGM_ALIVE_EXPIRED (see SWS WdgM for more details). Note that this value has to be recalculated with respect to the WdgM's own cycle time for ECU configuration.

Table 3.44: SupervisedEntityNeeds

Class	**SupervisedEntityCheckpointNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Watchdog Manager to support a Checkpoint for a Supervised Entity.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceNeeds			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.45: SupervisedEntityCheckpointNeeds

3.8.6 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 3.31.

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

[TPS_MANI_01052] 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] 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.

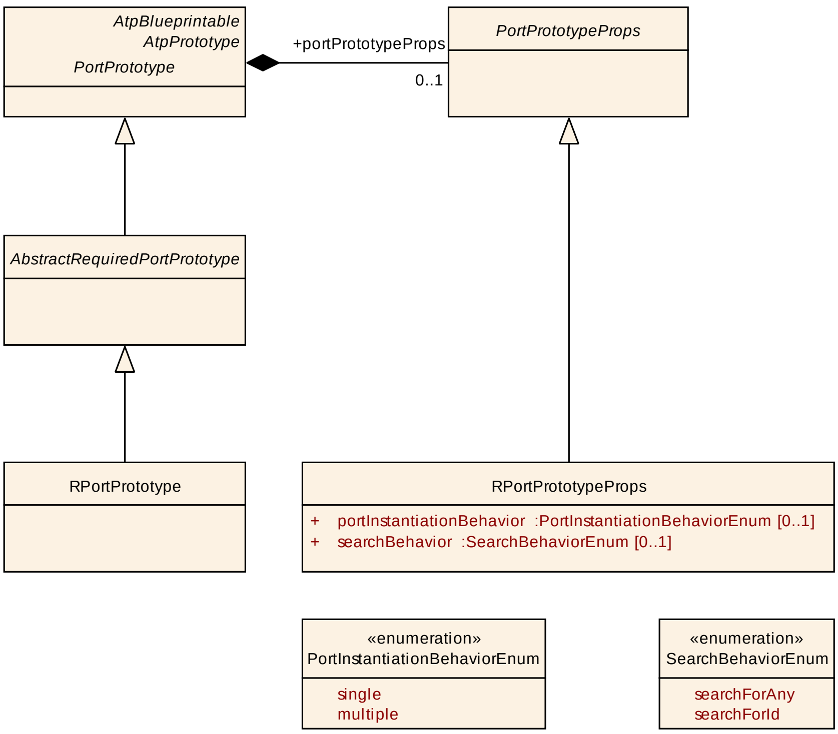

Figure 3.31: Modeling of the RPortPrototypeProps for RPortPrototype

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 PortPrototype. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 3.46: 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
portInstantiationBehavior	PortInstantiationBehaviorEnum	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 3.47: 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 3.48: 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
searchForId	This value represents the intention to search for a dedicated Id. **Tags:** atp.EnumerationValue=1

Table 3.49: SearchBehaviorEnum

3.8.7 Port Prototype ComSpec

[TPS_MANI_01053] 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 `ComSpec`s, i.e. specific sub-classes of the abstract meta-classes `RPortComSpec` and `PPortComSpec`.

However, the support for `ComSpec`s 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.

3.8.7.1 Port Prototypes typed by Service Interfaces

3.8.7.1.1 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" `event`s within the same `RPortPrototype` may need a different handling of the queue length.

[TPS_MANI_01054] 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`. ⌋*()*

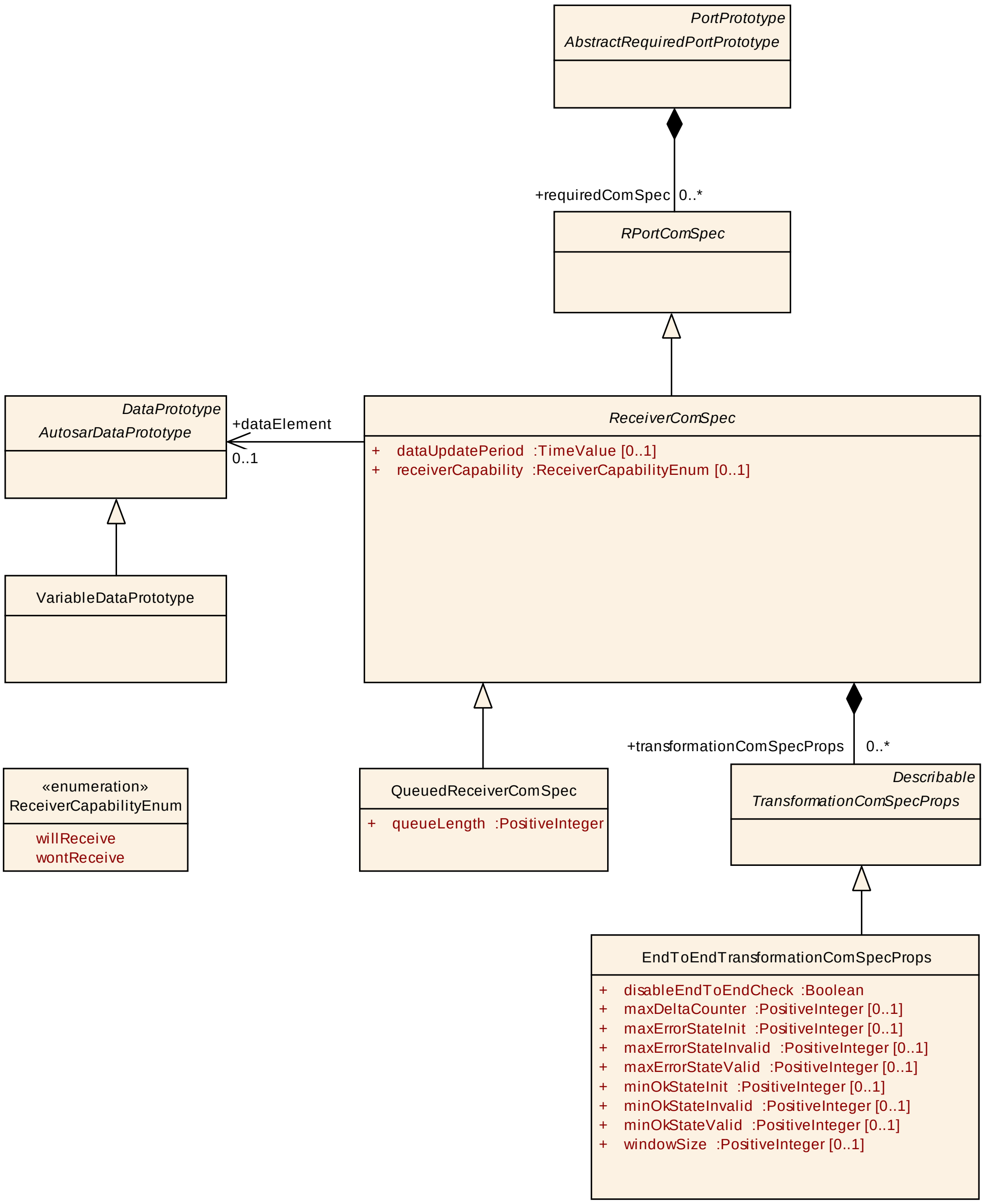

Figure 3.32: Modeling of the `ReceiverComSpec` on the *AUTOSAR adaptive platform*

Class	**ReceiverComSpec (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Receiver-specific communication attributes (RPortPrototype typed by SenderReceiverInterface).			
Base	ARObject, RPortComSpec			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
dataElement	AutosarDataPrototype	0..1	ref	Data element these attributes belong to.

dataUpdatePeriod	TimeValue	0..1	attr	This attribute defines the period in which the application shall check for updated data. This attribute is used for the configuration of the E2E protection. **Tags:** atp.Status=draft
receiverCapability	ReceiverCapabilityEnum	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
transformationComSpecProps	TransformationComSpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.50: 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 3.51: QueuedReceiverComSpec

[TPS_MANI_01106] 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 3.52: ReceiverCapabilityEnum

[TPS_MANI_03132] Semantics of E2E attributes in `ReceiverComSpec` ⌈ The `EndToEndTransformationComSpecProps` shall be used for the specification of `R-PortPrototype`-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 3.53: EndToEndTransformationComSpecProps

3.8.7.1.2 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.

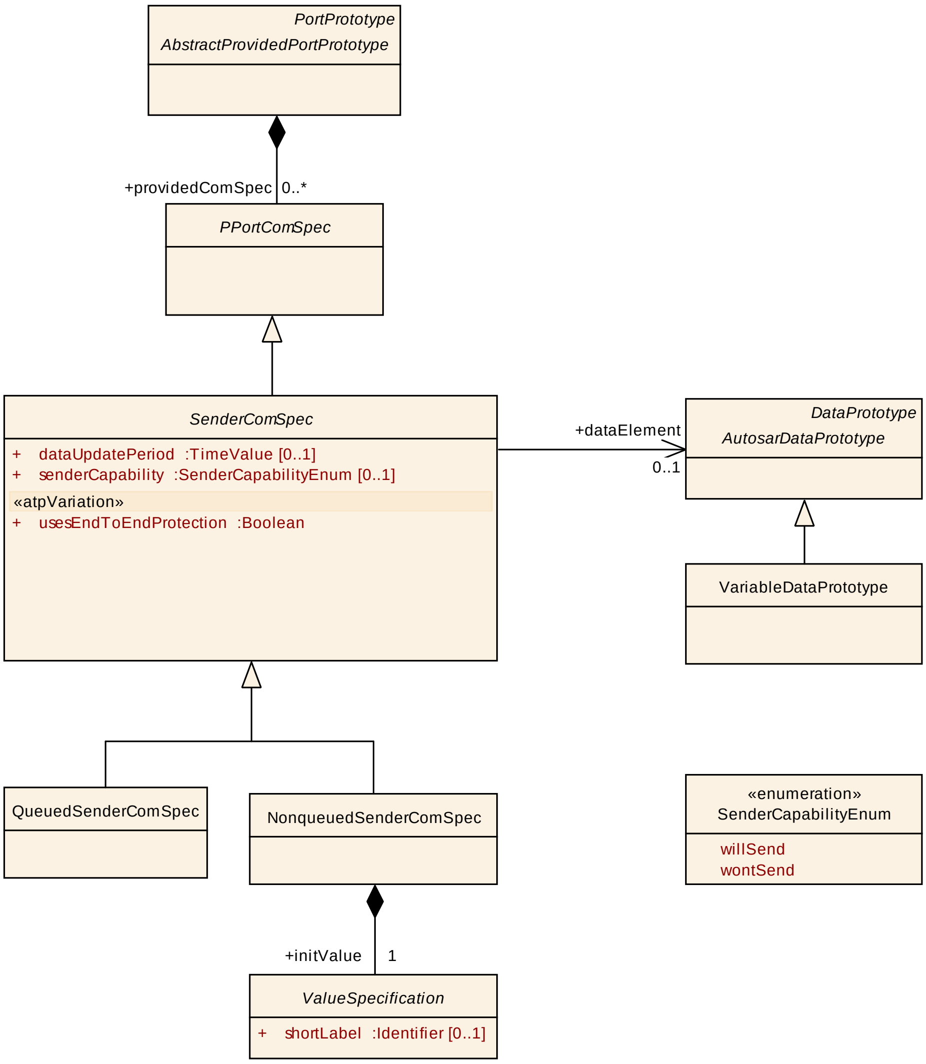

Figure 3.33: Modeling of the SenderComSpec on the *AUTOSAR adaptive platform*

Class	**SenderComSpec (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for a sender port (PPortPrototype typed by SenderReceiverInterface).			
Base	ARObject, PPortComSpec			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
compositeNetworkRepresentation	CompositeNetworkRepresentation	*	aggr	This represents a CompositeNetworkRepresentation defined in the context of a SenderComSpec.
dataElement	AutosarDataPrototype	0..1	ref	Data element these quality of service attributes apply to.
dataUpdatePeriod	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

networkRepresentation	SwDataDefProps	0..1	aggr	A networkRepresentation is used to define how the dataElement is mapped to a communication bus.
senderCapability	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
transmissionAcknowledge	TransmissionAcknowledgementRequest	0..1	aggr	Requested transmission acknowledgement for data element.
usesEndToEndProtection	Boolean	1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table 3.54: SenderComSpec

[TPS_MANI_01107] 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 `event`s 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 3.55: SenderCapabilityEnum

3.8.7.1.3 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.

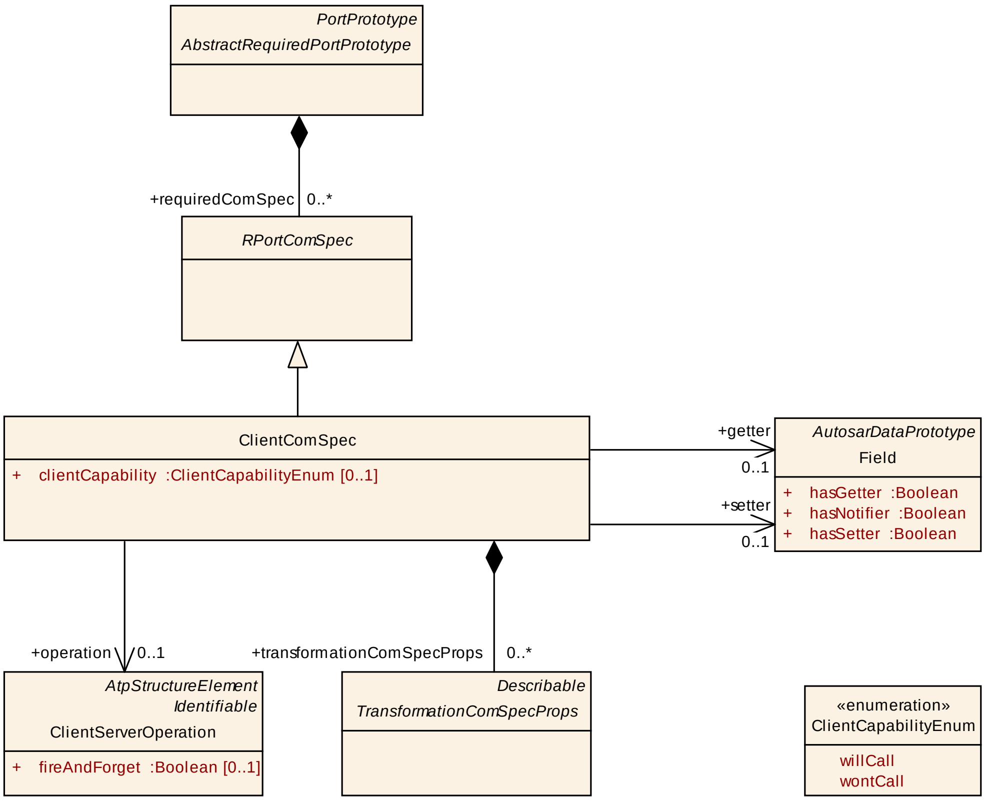

Figure 3.34: Modeling of the `ClientComSpec` on the *AUTOSAR adaptive platform*

Class	**ClientComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Client-specific communication attributes (RPortPrototype typed by ClientServerInterface).			
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 ClientComSpec 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 ClientComSpec refers to the setter of a Field. **Tags:** atp.Status=draft

transformationComSpecProps	TransformationComSpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.56: ClientComSpec

[TPS_MANI_01108] 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` resp. 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 3.57: 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*.

3.8.7.2 Port Prototypes typed by Persistency Data Interfaces

[TPS_MANI_01069] 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 `PersistencyProvidedComSpec` is provided. ⌋ *(RS_MANI_00027)*

[TPS_MANI_01074] Specification of encryption of persistent data ⌈ The specification that data related to a specific `PortPrototype` typed by a specific `PersistencyKeyValueDatabaseInterface` shall be encrypted can be made by having a `CryptoNeedToPortPrototypeMapping` refer to the mentioned `PortPrototype`. ⌋*(RS_MANI_00027)*

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 PersistencyProvidedComSpec.

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_01075] Specification of redundancy of persistent data ⌈ The attribute PersistencyProvidedComSpec.redundancy can be taken to specify whether the respective key-value database shall store data redundantly from the perspective of the designer of the software-component. ⌋*(RS_MANI_00027)*

In contrast to the definition of encryption the specification of redundancy doesn't leave much freedom for the designer of a software-component. This person may only state that the values in the corresponding key-value database shall be or shall not be stored redundantly.

The details are left to an integrator who may also decide to overrule the value of PersistencyProvidedComSpec.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 the behavior of a given persistent data element with respect to redundancy. **Tags:** atp.Status=draft
Literal	***Description***
none	This value represents the requirement that a piece of data to be stored persistently shall not end up in a redundant persistent storage facility. **Tags:** atp.EnumerationValue=1
redundant	This value represents the requirement that a piece of data to be stored persistently shall end up in a redundant persistent storage facility. The nature of the redundant persistent storage is not further qualified and subject to integrator decisions. **Tags:** atp.EnumerationValue=0

Table 3.58: PersistencyRedundancyEnum

3.9 Adaptive AUTOSAR Application

This section contains the description of the formal modeling of the concept of an "application" itself. For this purpose, the meta-class AdaptiveAutosarApplication has been created.

[TPS_MANI_01008] Semantics of `AdaptiveAutosarApplication` ⌈ Meta-class `AdaptiveAutosarApplication` represents the unit of distribution of application software for the adaptive platform towards an integration step, i.e. application software developers shall pass the results of their work in the form of an `AdaptiveAutosarApplication` to the integration workflow. ⌋*(RS_MANI_00001)*

Class	**AdaptiveAutosarApplication**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This element describes a collection of executables that forms an Adaptive AUTOSAR Application. This corresponds to the definition of Application in SWS Execution Management. **Tags:** atp.Status=draft; atp.recommendedPackage=AdaptiveAutosarApplications			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
executable	Executable	1..*	ref	Reference to executables that are contained in the Adaptive Autosar Application. **Tags:** atp.Status=draft
version	String	0..1	attr	Version of the Adaptive Autosar Application

Table 3.59: AdaptiveAutosarApplication

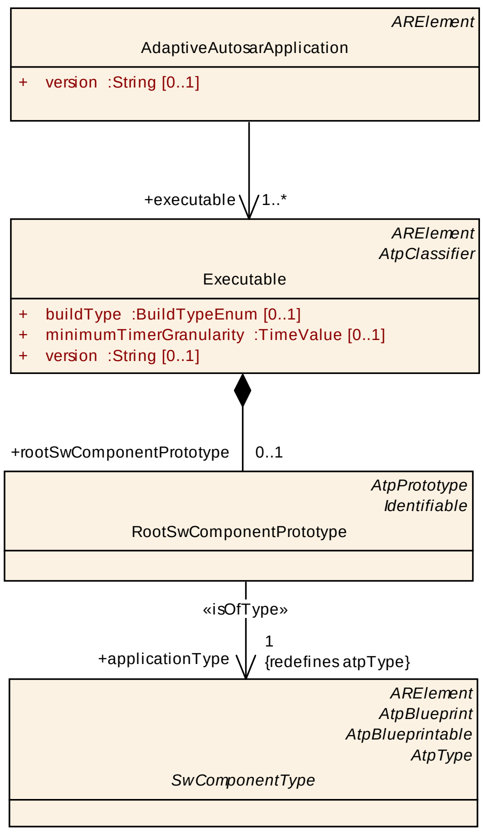

Figure 3.35: Modeling of the `AdaptiveAutosarApplication` and `Executable`

In general, an AdaptiveAutosarApplication may not be limited to the actual application level (i.e. conceptually located *above* the middleware), it is also supported to define an AdaptiveAutosarApplication that actually represents a part of the concrete implementation of an *AUTOSAR adaptive platform*.

A possible example for this kind of application could be a Diagnostic Manager (DM).

[TPS_MANI_01009] Standardized values of AdaptiveAutosarApplication.category ⌈ The following values of attribute AdaptiveAutosarApplication.category are standardized by AUTOSAR:

- APPLICATION_LEVEL: the AdaptiveAutosarApplication represents software on the application level (i.e. conceptually located *above* the middleware).
- PLATFORM_LEVEL: the AdaptiveAutosarApplication represents software on the platform level (i.e. conceptually located *on the level of* the middleware).

⌋*(RS_MANI_00001)*

Both the meta-class AdaptiveAutosarApplication and the meta-class Executable provide the ability to define a version.

The format and content of these version specifications is not constrained by the AUTOSAR standard, i.e the content of attribute version can be defined in custom ways and the AUTOSAR standard does **not** make any assumptions on how different values of version are compared to each other.

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.
minimumTimerGranularity	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
rootSwComponentPrototype	RootSwComponentPrototype	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

transformationPropsMappingSet	TransformationPropsToServiceInterfaceElementMappingSet	0..1	ref	Reference to a set of serialization properties that are defined for ServiceInterfaces of the Executable. **Tags:** atp.Status=draft
version	String	0..1	attr	Version of the executable. **Tags:** atp.Status=draft

Table 3.60: Executable

Enumeration	**BuildTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::AdaptiveModuleImplementation
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 3.61: BuildTypeEnum

Each `AdaptiveAutosarApplication` can refer to 1..* `Executable`s. For practical purposes, this relation can be translated to "`AdaptiveAutosarApplication` consists of 1..* `Executable`s.

In contrast to a potential modeling of this relation as an aggregation, however, the reference-based approach supports the existence of the same `Executable` in the collection `AdaptiveAutosarApplication.executable` of several `AdaptiveAutosarApplication`s.

[TPS_MANI_01010] 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*.

[TPS_MANI_03056] 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] `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 3.36 exemplifies the statement of [constr_1492]. The example shows a component hierarchy that consists of `SwComponentPrototype`s that are excursively typed by either a `CompositionSwComponentType` or an `AdaptiveApplicationSwComponentType`.

While the left part of Figure 3.36 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.

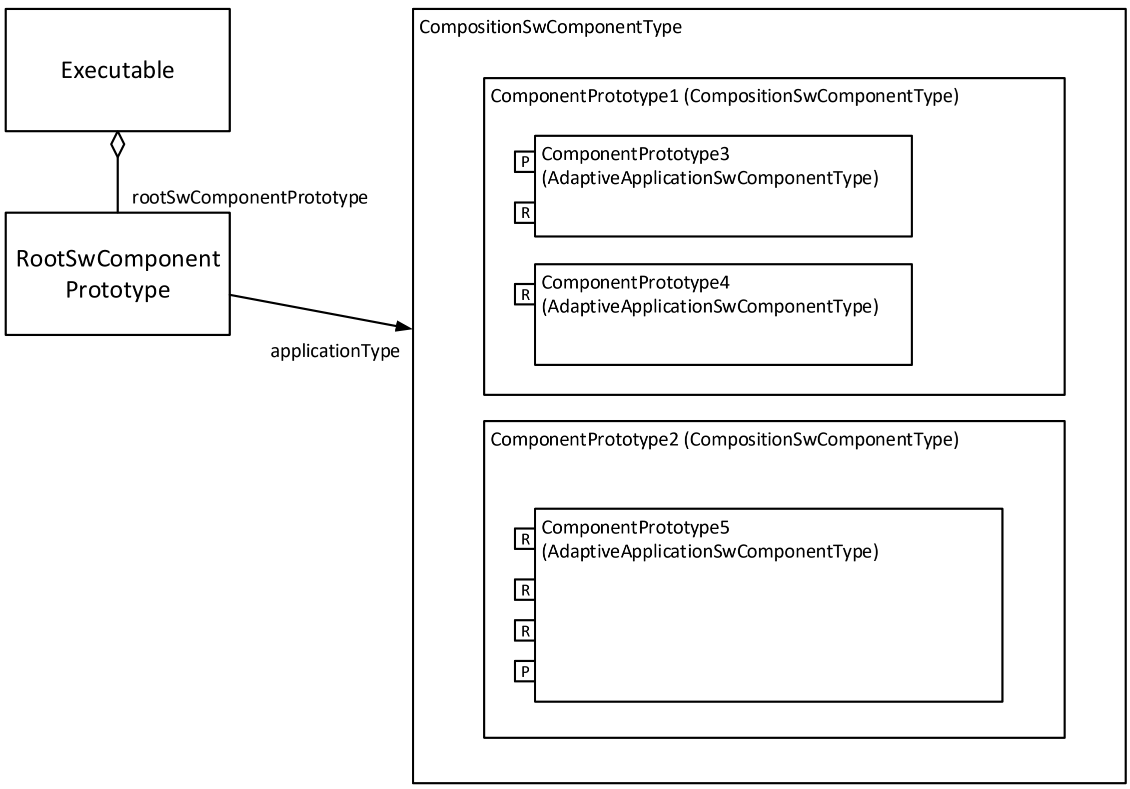

Figure 3.36: Example of the possible structure of an `Executable`

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 a `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 PortPrototypes, PortInterfaces, VariableDataPrototypes, etc.). **Tags:** atp.Status=draft			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
applicationType	SwComponentType	1	tref	This SwComponnetType acts as the Type of the RootSwComponentPrototype. **Stereotypes:** isOfType **Tags:** atp.Status=draft

Table 3.62: 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			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

consistencyNeeds	ConsistencyNeeds	*	aggr	This represents the collection of ConsistencyNeeds owned by the enclosing SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
port	PortPrototype	*	aggr	The PortPrototypes through which this SwComponentType 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
swComponentDocumentation	SwComponentDocumentation	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
unitGroup	UnitGroup	*	ref	This allows for the specification of which UnitGroups are relevant in the context of referencing SwComponentType.

Table 3.63: SwComponentType

Class	**CompositionSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by SwComponentTypes) as well as SwConnectors for primarily connecting SwComponentPrototypes among each others and towards the surface of the CompositionSwComponentType. By this means hierarchical structures of software-components can be created. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

component	SwComponentPrototype	*	aggr	The instantiated components that are part of this composition. The aggregation of SwComponentPrototype is subject to variability with the purpose to support the conditional existence of a SwComponentPrototype. Please be aware: if the conditional existence of SwComponentPrototypes is resolved post-build the deselected SwComponentPrototypes are still contained in the ECUs build but the instances are inactive in in that they are not scheduled by the RTE. The aggregation is marked as atpSplitable in order to allow the addition of service components to the ECU extract during the ECU integration. The use case for having 0 components owned by the CompositionSwComponentType could be to deliver an empty CompositionSwComponentType to e.g. a supplier for filling the internal structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses. The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow. The aggregation is marked as atpSplitable in order to allow the extension of the ECU extract with AssemblySwConnectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
constantValueMapping	ConstantSpecificationMappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortComSpec. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=constantValueMapping

dataTypeMapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the used ApplicationDataTypes in PortInterfaces. Background: when developing subsystems it may happen that ApplicationDataTypes are used on the surface of CompositionSwComponentTypes. In this case it would be reasonable to be able to also provide the intended mapping to the ImplementationDataTypes. However, this mapping shall be informal and not technically binding for the implementers mainly because the RTE generator is not concerned about the CompositionSwComponentTypes. Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the mapping to ImplementationDataTypes is not impacting compatibility. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dataTypeMapping
instantiationRTEEventProps	InstantiationRTEEventProps	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortLabel, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

Table 3.64: CompositionSwComponentType

3.10 Optional Members in complex Data Structures

3.10.1 Background

The *AUTOSAR adaptive platform* supports the usage of a `TLV`[5] 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.

[5] This abbreviation stands for tag-length-value

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.

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

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

On top of that, 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.

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 ServiceInterface.

3.10.2 Definition of Optionality

As mentioned before, the concrete definition of optionality on the level of a ServiceInterface is done by the indication of individual DataPrototypes that are elements of a composite data structure as optional.

[TPS_MANI_01082] Eligibility of DataPrototypes for the definition of optionality
⌈DataPrototypes identified as optional can only exist as

- part of a **composite data type** of category STRUCTURE
- arguments of a method within the scope of a ServiceInterface

⌋*(RS_MANI_00030)*

In other words, there is one use case (i.e. optional method argument) to define entire elements of a `ServiceInterface` as optional.

With respect to the question of eligibility for the definition of optional elements in array data structures: there is already a mechanism in place for the definition of variable-size arrays (VSA) described in the context of AUTOSAR and it would not make sense to add another way to achieve the same semantics.

The details are explained in the specification of the AUTOSAR Software Component Template [1].

The concrete modeling approach for the definition of optionality in the context of a `ServiceInterface` is sketched in Figure 3.37. The anchor point for this effort is the `ServiceInterfaceSubElement`.

[TPS_MANI_01083] Optionality is supported for `ApplicationDataType` as well as `ImplementationDataType` ⌈ The `ServiceInterfaceSubElement` supports the definition of optionality for the case the respective `DataPrototype` is typed by either an `ApplicationDataType` or an `ImplementationDataType`. ⌋ *(RS_MANI_00030)*

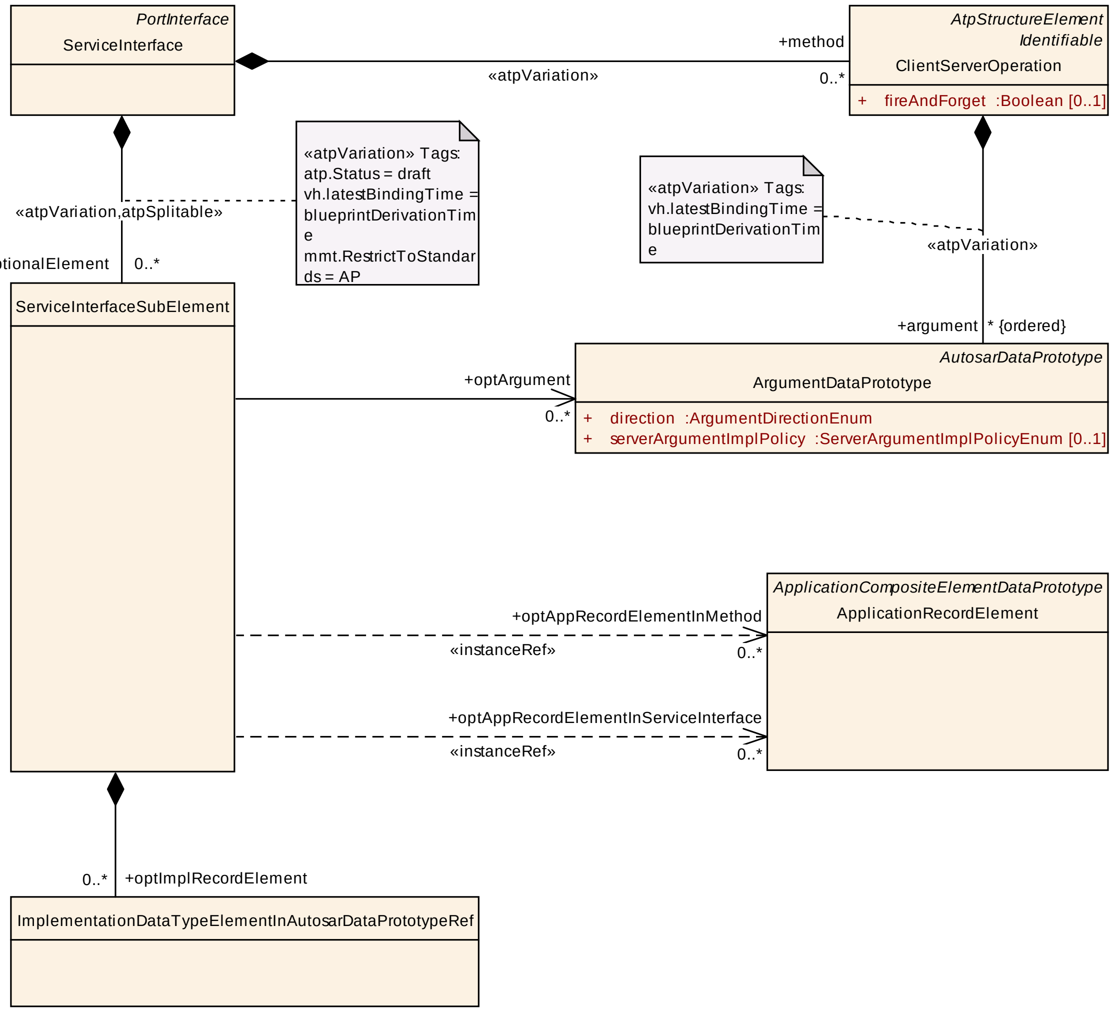

Figure 3.37: Modeling of optional members of DataPrototypes in the context of a ServiceInterface

[TPS_MANI_01084] Optionality for a DataPrototype typed by an ApplicationDataType ⌈ A DataPrototype typed by an ApplicationDataType that is eligible for the definition of optionality can only be an ApplicationRecordElement or an entire ArgumentDataPrototype. ⌋*(RS_MANI_00030)*

[TPS_MANI_01133] Optional element of an event ⌈ An ApplicationRecordElement can in principle be used inside the definition of an event or field. In this case the reference ServiceInterfaceSubElement.optAppRecordElementInServiceInterface shall exist. ⌋*(RS_MANI_00030)*

[TPS_MANI_01134] Optional element in the context of a method ⌈ If optionality occurs in the context of a ClientServerOperation then one of the following cases applies:

- An entire argument identified as optional shall be referenced in the role ServiceInterfaceSubElement.optArgument.

- An element of an argument typed by a composite data type identified as optional shall be referenced in the role ServiceInterfaceSubElement.optAppRecordElementInMethod.

⌋(RS_MANI_00030)

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**
–	–	–	–	–

Table 3.65: ApplicationRecordElement

Please note that the existence of the two different instanceRefs (in the roles optAppRecordElementInServiceInterface and optAppRecordElementInMethod) to ApplicationRecordElement for the same purpose of defining optionality has a purely formal background.

In more technical terms, the first atpContextElement of the reference optAppRecordElementInServiceInterface is the ServiceInterface while in the case of the reference optAppRecordElementInMethod the first atpContextElement is the ClientServerOperation.

For details regarding this formal aspect, please consult the definition of the "abstract structure" in the specification of the AUTOSAR Generic Structure Template [5].

More details about the specific modeling of ServiceInterfaceSubElement.optAppRecordElementInServiceInterface and ServiceInterfaceSubElement.optAppRecordElementInMethod can be found in section B.

Class	**ServiceInterfaceSubElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::OptionalElementInServiceInterface			
Note	This meta-class represents the ability to refer to sub-elements of fields, events, and arguments to methods that are typed by a composite data type **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
optAppRecordElementInMethod	ApplicationRecordElement	*	iref	This reference identifies an element of an ApplicationRecordDataType as an optional sub-element in the context of an argument to a method defined in the context of an enclosing ServiceInterface. **Tags:** atp.Status=draft

optAppRecordElementInServiceInterface	ApplicationRecordElement	*	iref	This reference identifies an element of an ApplicationRecordDataType as an optional sub-element of an event or a field in the context of the enclosing ServiceInterface. **Tags:** atp.Status=draft
optArgument	ArgumentDataPrototype	*	ref	This reference identifies optional arguments in the context of a ClientServerOperation. **Tags:** atp.Status=draft
optImplRecordElement	ImplementationDataTypeElementInAutosarDataPrototypeRef	*	aggr	This aggregation provides the ability to refer to ImplementationDataTypeElements as optional elements in the contect of a ServiceInterface. **Tags:** atp.Status=draft

Table 3.66: ServiceInterfaceSubElement

Please note that the usage of `ApplicationDataType`s for the specification of a `ServiceInterface` for which optionality is defined has the implication that the code generator that creates the `API` towards the application software needs to take the optionality into account in the structurally identical `ImplementationDataType` that is tied to the respective `ApplicationDataType` by means of the applicable `DataTypeMap`.

[TPS_MANI_01085] Definition of optionality for a `DataPrototype` typed by an `ImplementationDataType` ⌈ For the definition of optionality for a `DataPrototype` typed by an `ImplementationDataType` it is necessary to aggregate `ImplementationDataTypeElementInAutosarDataPrototypeRef` in the role `optImplRecordElement` at the `ServiceInterfaceSubElement`. ⌋*(RS_MANI_00030)*

Nevertheless, the `ImplementationDataTypeElement` finally referenced as the target element in the context of an `ImplementationDataTypeElementInAutosarDataPrototypeRef` shall be part of a record data type.

[constr_1527] `ImplementationDataTypeElement` finally referenced as the target element in the context of an `ImplementationDataTypeElementInAutosarDataPrototypeRef` ⌈ An `ImplementationDataTypeElement` referenced in the role `ImplementationDataTypeElementInAutosarDataPrototypeRef.targetDataPrototype` shall be aggregated by either of the following options:

1. An `ImplementationDataType` of `category` STRUCTURE.
2. An `ImplementationDataTypeElement` of `category` STRUCTURE .

⌋*()*

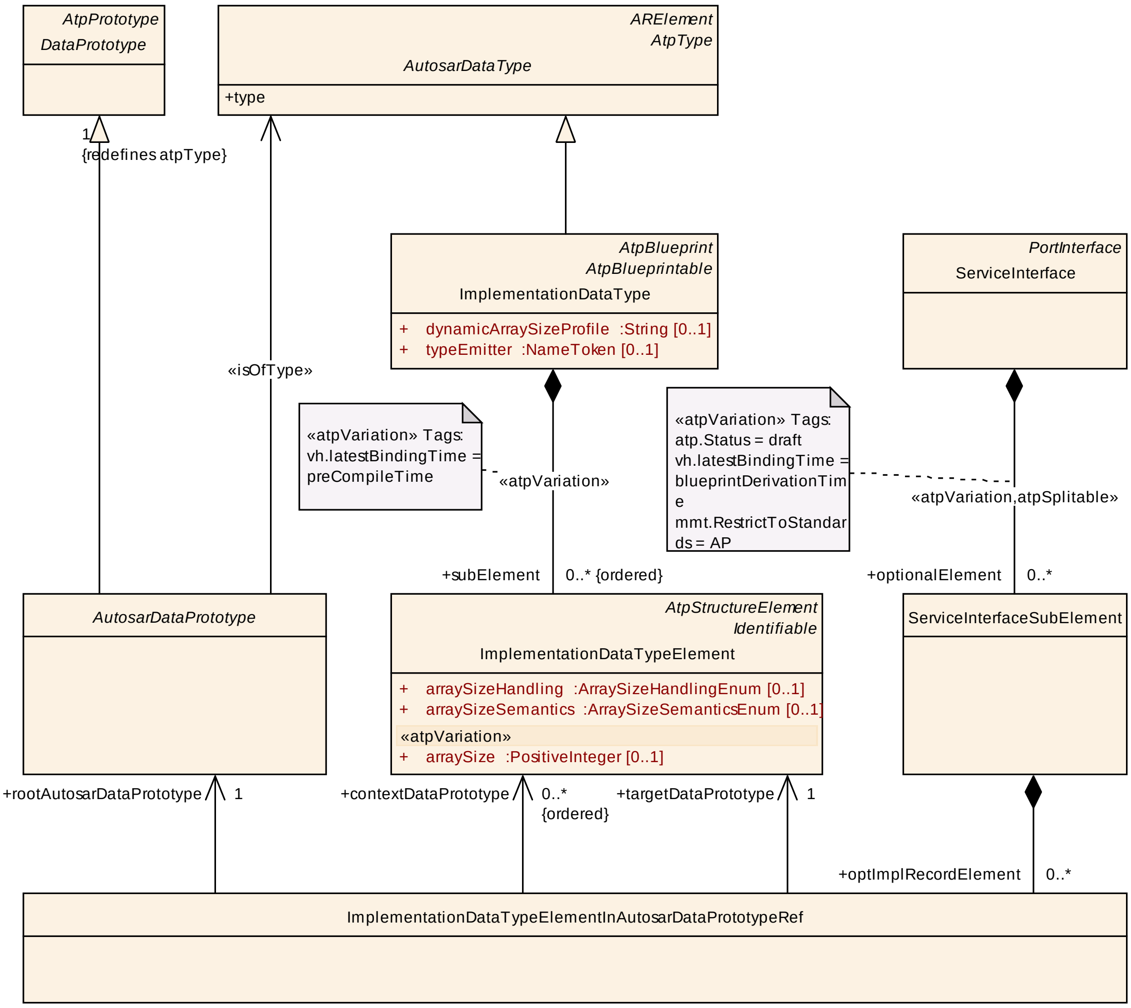

Figure 3.38: Modeling of optional members of `DataPrototype`s typed by `ImplementationDataType` in the context of a `ServiceInterface`

Class	**ImplementationDataTypeElementInAutosarDataPrototypeRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::OptionalElementIn ServiceInterface			
Note	This meta-class represents the ability to refer to an ImplementationDataTypeElement in the context of a ServiceInterface. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
contextDataPrototype (ordered)	ImplementationDataTypeElement	*	ref	This reference goes to context DataPrototypes. It is only required if indirections in the definition of the respective ImplementationDataType exist. **Tags:** atp.Status=draft xml.sequenceOffset=20

rootAutosarDataPrototype	AutosarDataPrototype	1	ref	This reference goes to either the event, field, or method argument in the role of a root element of a composite data structure typed by an ImplementationDataType. **Tags:** atp.Status=draft xml.sequenceOffset=10
targetDataPrototype	ImplementationDataTypeElement	1	ref	This reference points to the target ImplementationDataElement inside a composite ImplementationDataType **Tags:** atp.Status=draft xml.sequenceOffset=30

Table 3.67: ImplementationDataTypeElementInAutosarDataPrototypeRef

As mentioned before, a limitation in terms of data types used for the definition of optionality applies.

The nature of the limitation is that - within the context of one ServiceInterface - optionality shall be uniformly defined, i.e. all affected DataPrototypes shall not differ in terms of how they are referenced from a ServiceInterfaceSubElement.

[constr_1528] Definition of optionality for multiple DataPrototypes typed by the same AutosarDataType ⌈ Within the context of a given ServiceInterface, for each data type that has child elements that are referenced directly (if the DataPrototype is typed by an ApplicationDataType) or indirectly (if the DataPrototype is typed by an ImplementationDataType) from a ServiceInterfaceSubElement it is required that every DataPrototype typed by the respective composite AutosarDataType shall have exactly the same (in terms of the target DataPrototype within the composite DataPrototype) set of references in the role ServiceInterface.optionalElement. ⌋*()*

Figure 3.39 contains a simplified sketch of the main statement of [constr_1528]. Of the three ServiceInterfaceSubElement, two define a consistent reference to one DataPrototype as a sub-element of the composite data type.

The third ServiceInterfaceSubElement refers to a different DataPrototype within the composite data type. In this case a violation of [constr_1528] shall be reported.

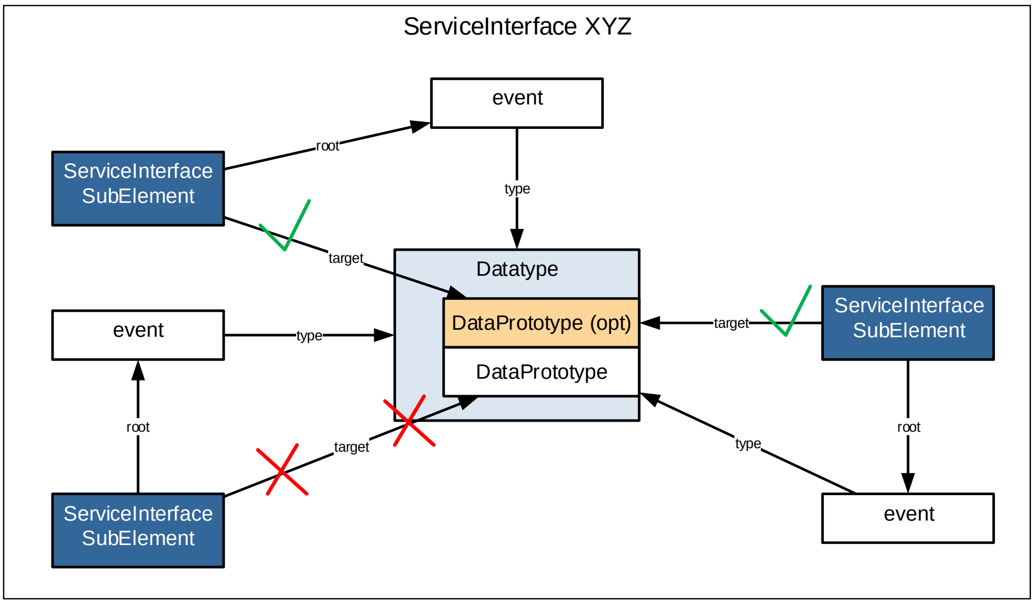

Figure 3.39: Simplified example of the application of ServiceInterfaceSubElement

Please note that the modeling approach for the definition of optionality on the level of a ServiceInterface explicitly leaves out the question of how the necessary tag id is assigned.

It is assumed that the step in the project workflow where ServiceInterfaces and their optional elements are defined is not the step where the assignment of an actual TLV data id (that is not to be confused with the actual tag that is used on the bus to identify the element) is in the focus.

Therefore, the assignment of TLV data ids has intentionally been separated from the rest of the definition of optionality. Details can be found in section 3.11.3.

3.11 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 in the context of an Executable that is valid for all available occurrences of a DataPrototype in the ServiceInterface element. This case is described in detail in chapter 3.11.1.
- Fine granular modeling on the level of DataPrototypes described in this chapter. This case is described in detail in chapter 3.11.2.

3.11.1 Default Values for Serialization Properties

[TPS_MANI_03101] 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] 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] Default size for all array 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 arrays defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field. ⌋*(RS_MANI_00008, RS_MANI_00025)*

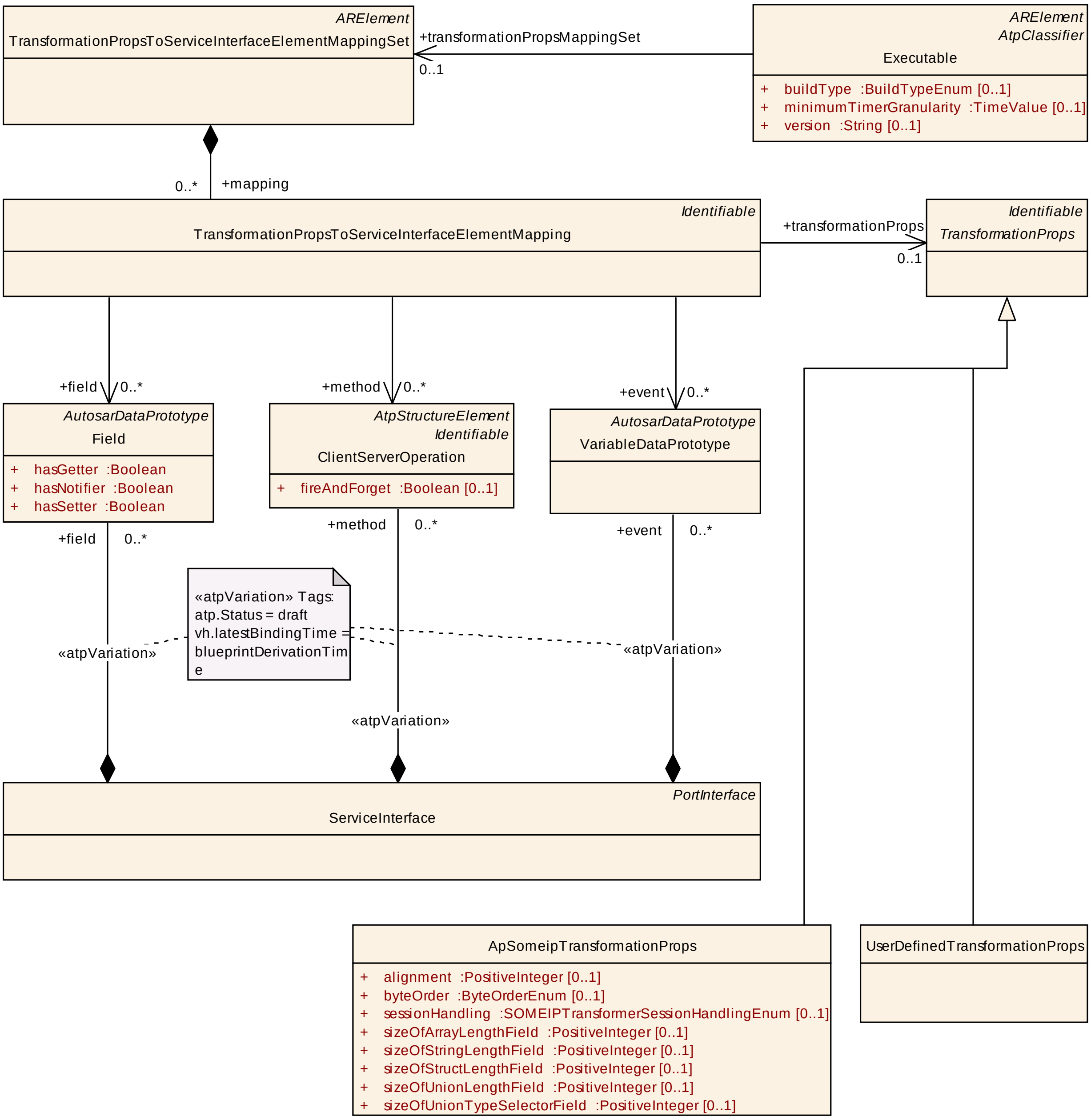

Figure 3.40: Association of serialization properties with a ServiceInterface in the context of an Executable

[TPS_MANI_03104] 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] 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] 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] 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] 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] 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)

Please note that more details about ApSomeipTransformationProps can be found in chapter 3.11.2.

Class	**ApSomeipTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
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.
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 an Array. It describes the size of the length field (in Bytes) that will be put in front of the Array in the SOME/IP message. In contrast to Classic AUTOSAR this attribute defines the value for both, fixed-size and dynamic-size arrays.
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.

Table 3.68: ApSomeipTransformationProps

[TPS_MANI_03102] 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 [5].

Class	TransformationPropsToServiceInterfaceElementMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
Note	Collection of TransformationPropsToServiceInterfaceElementMappings. **Tags:** atp.Status=draft; atp.recommendedPackage=TransformationPropsToServiceInterfaceMappingSets			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Type	Mul.	Kind	Note
mapping	TransformationPropsToServiceInterfaceElementMapping	*	aggr	Mapping that assigns serialization properties to elements of a ServiceInterface. **Tags:** atp.Status=draft

Table 3.69: 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
transformationProps	TransformationProps	0..1	ref	This represents the reference to the applicable Serialization properties. **Tags:** atp.Status=draft

Table 3.70: TransformationPropsToServiceInterfaceElementMapping

Class	UserDefinedTransformationProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
Note	UserDefined serialization properties. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, TransformationProps			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 3.71: UserDefinedTransformationProps

3.11.2 Individual Definition of Serialization Properties

[TPS_MANI_03109] `TransformationProps` on the level of `DataPrototype`s overwrites `TransformationProps` settings on the level of a `ServiceInterface` ⌈ The fine granular modeling of `TransformationProps` on the level of `DataPrototype`s overwrites the `TransformationProps` settings defined on the level of a `ServiceInterface` described with the `TransformationPropsToServiceInterfaceElementMappingSet`. ⌋*()*

[constr_3361] 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 `DataPrototype`s of the `ServiceInterface` element as well. ⌋*()*

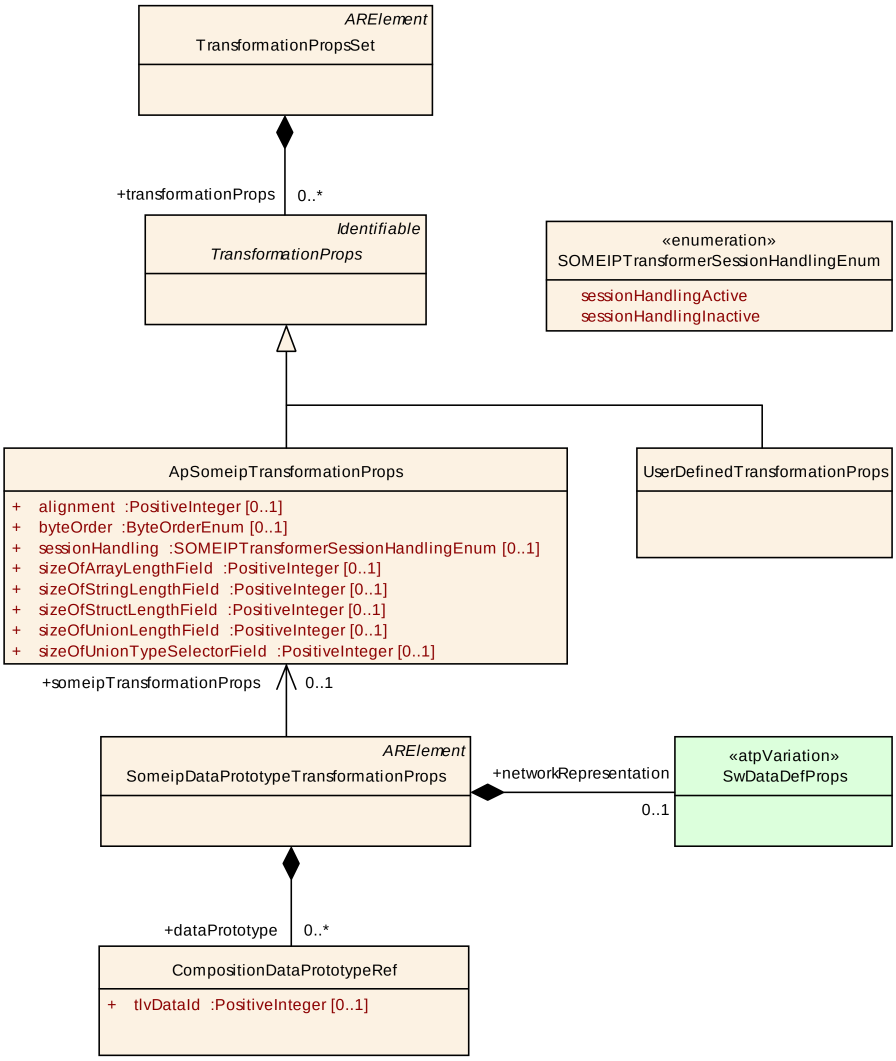

Figure 3.41: Overview about SOME/IP Serialization Properties

Class	**ApSomeipTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
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.
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 an Array. It describes the size of the length field (in Bytes) that will be put in front of the Array in the SOME/IP message. In contrast to Classic AUTOSAR this attribute defines the value for both, fixed-size and dynamic-size arrays.
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.

Table 3.72: ApSomeipTransformationProps

Enumeration	**SOMEIPTransformerSessionHandlingEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer
Note	Enables or disable session handling for SOME/IP transformer
Literal	***Description***
sessionHandlingActive	The SOME/IP Transformer shall use session handling **Tags:** atp.EnumerationValue=0
sessionHandlingInactive	The SOME/IP Transformer doesn't use session handling **Tags:** atp.EnumerationValue=1

Table 3.73: SOMEIPTransformerSessionHandlingEnum

[TPS_MANI_03070] Size of a length field for a chosen array ⌈ The attribute `sizeOfArrayLengthField` of `ApSomeipTransformationProps` defines the size of a length field generated by SOME/IP in front of an array for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the array that is referenced within the aggregated `CompositionDataPrototypeRef`. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3353] Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLengthField ⌈ The value of the attribute sizeOfArrayLengthField shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03071] 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 CompositionDataPrototypeRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3354] Restriction in usage of ApSomeipTransformationProps.sizeOfStructLengthField ⌈ The value of the attribute sizeOfStructLengthField shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03116] 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 CompositionDataPrototypeRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3372] Restriction in usage of ApSomeipTransformationProps.sizeOfStringLengthField ⌈ The value of the attribute sizeOfStringLengthField shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03072] 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 CompositionDataPrototypeRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3355] Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLengthField ⌈ The value of the attribute sizeOfUnionLengthField shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03073] 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 CompositionDataPrototypeRef. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3356] Restriction in usage of ApSomeipTransformationProps.alignment ⌈ The value of the attribute alignment shall always be divisible by 8. ⌋*()*

[TPS_MANI_03074] 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 CompositionDataPrototypeRef. ⌋ *(RS_MANI_00008, RS_MANI_00024)*

[constr_3357] Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField ⌈ The value of the attribute sizeOfUnionTypeSelectorField shall be either 1, 2 or 4. ⌋ *()*

[TPS_MANI_03075] 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 CompositionDataPrototypeRef. ⌋ *(RS_MANI_00008, RS_MANI_00024)*

Class	**SomeipDataPrototypeTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
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=SomeipDataPrototype TransformationPropss			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataPrototype	CompositionDataPrototypeRef	*	aggr	Collection of DataPrototypes for which the settings in SomeipDataPrototypeTransformationProps are valid. For reuse reasons the SomeipDataPrototypeTransformationProps is able to aggregate several DataPrototypes. **Tags:** atp.Status=draft
networkRepresentation	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 networkRepresentation is not provided then the baseType of the ImplementationDataType shall be used for the serialization/deserialization. **Tags:** atp.Status=draft
someipTransformationProps	ApSomeipTransformationProps	0..1	ref	This reference represents the ability to define data transformation props specifically for a SOME/IP serialization. **Tags:** atp.Status=draft

Table 3.74: SomeipDataPrototypeTransformationProps

Class	**CompositionDataPrototypeRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to refer to an AUTOSAR DataPrototype in the context of a CompositionSwComponentType. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataPrototype	DataPrototype	0..1	iref	This attribute shall exist if the InstanceRef points to a DataPrototype typed by an ApplicationDataType. **Tags:** atp.Status=draft
elementInImplDatatype	ElementInImplementationDatatypeInstanceRef	0..1	aggr	This attribute shall exist if the InstanceRef points to a DataPrototype typed by an ImplementationDataType. **Tags:** atp.Status=draft
tlvDataId	PositiveInteger	0..1	attr	This attribute represents the ability to specify a TLV data-id for the serialization of a specific DataPrototype in the context of a (potentially deeply-nested) composite data structure for the case that the data structure has optional elements. This value does not represent the entire value of the tag, e.g. the wire-type is not included (because it can be derived from the information about the underlying AutosarDataType).

Table 3.75: CompositionDataPrototypeRef

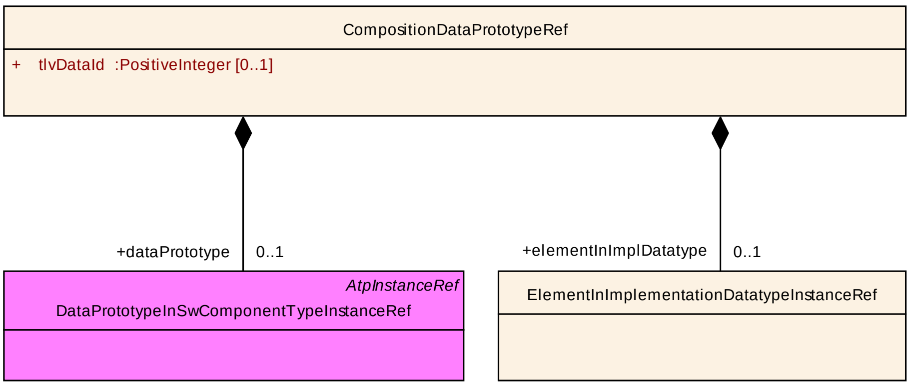

Figure 3.42: Reference to a DataPrototype in the context of a CompositionSwComponentType that is typed by an ApplicationDataType or by an ImplementationDataType

[TPS_MANI_01132] Semantics of `CompositionDataPrototypeRef` ⌈ If the target `DataPrototype` inside the `CompositionDataPrototypeRef` is typed by an `ApplicationDataType` then the reference `CompositionDataPrototypeRef.dataPrototype` is sufficient.

If the target DataPrototype is typed by an ImplementationDataType and if the target DataPrototype is a member of a complex data type then the reference CompositionDataPrototypeRef.dataPrototype and the reference CompositionDataPrototypeRef.elementInImplDatatype shall exist.

In this case the reference CompositionDataPrototypeRef.dataPrototype identifies the AutosarDataPrototype in the context of a PortPrototype and the rest (i.e. pointing into the AutosarDataPrototype) is done by means of the reference CompositionDataPrototypeRef.elementInImplDatatype. ⌋ *(RS_MANI_00008, RS_MANI_00024)*

In other words, the reference to an AutosarDataPrototype typed by an ImplementationDataType still requires only the existence of CompositionDataPrototypeRef.dataPrototype.

The usage of the SomeipDataPrototypeTransformationProps.networkRepresentation is explained in more detail in the System Template [10] in [TPS_SYST_02136] and [TPS_SYST_02137].

3.11.3 Assignment of TLV Data IDs for Data Structures with optional Members

[TPS_MANI_01097] Assignment of TLV data ids for data structures with optional members ⌈ The assignment of TLV data ids for data structures with optional members is done in the context of the specification of SomeipDataPrototypeTransformationProps, namely by means of the attribute SomeipDataPrototypeTransformationProps.dataPrototype.tlvDataId. ⌋ *(RS_MANI_00030)*

This approach takes benefit from the fact that the CompositionDataPrototypeRef is able to create references into any nested level of data structures.

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 CompositionDataPrototypeRef.tlvDataId.

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 [11]) is the motivation for the existence of [constr_1532].

[constr_1532] Consistent assignment of TLV data ids to data structures with optional members ⌈ For every DataPrototype or ImplementationDataType resp. ImplementationDataTypeElement of category STRUCTURE where direct members are the target of either

- ServiceInterfaceSubElement.optAppRecordElementInMethod
- ServiceInterfaceSubElement.optAppRecordElementInServiceInterface

- ServiceInterfaceSubElement.optImplRecordElement

references to **all direct members** of this DataPrototype or ImplementationDataType resp. ImplementationDataTypeElement shall be created on the basis of the definition of CompositionDataPrototypeRef and **within the definition of each** respective CompositionDataPrototypeRef the attribute CompositionDataPrototypeRef.tlvDataId **shall exist and have a unique value** in the context of respective enclosing DataPrototype or ImplementationDataType resp. ImplementationDataTypeElement. ⌋*()*

Please note, however, that [constr_1532] only extends to the existence of CompositionDataPrototypeRef.tlvDataId.

The numerical values of CompositionDataPrototypeRef.tlvDataId for eligible DataPrototypes or ImplementationDataTypeElements within the context of the same ApplicationRecordDataType resp. ImplementationDataType of category STRUCTURE don't have to be identical.

In other words, if the ids for the elements of a given structure are set to 1, 2, and 3 for one DataPrototype then it is possible that the ids for another DataPrototype typed by the same ApplicationRecordDataType are set to different values, e.g. 4, 5, and 6.

To bring this example one step further, the definition of values 1, 1, and 3 as tag ids in the context of the enclosing DataPrototype **would have to be rejected** by regulation of [constr_1532].

The important aspect of the definition of tag ids (similar to other serialization-related information) is that the settings need to be shared between sender and receiver.

Obviously, a constraint similar to [constr_1532] (and similarly motivated by the existence of [PRS_SOMEIP_00231], see [11]) needs to exist for the case of optional arguments in a ClientServerOperation.

[constr_1537] Consistent assignment of TLV data ids to arguments of a given ClientServerOperation ⌈ For each ClientServerOperation where at least one argument is the target of a reference in the role ServiceInterfaceSubElement.optArgument references to all arguments shall be created on the basis of the definition of CompositionDataPrototypeRef and **within the definition of each** respective CompositionDataPrototypeRef the attribute CompositionDataPrototypeRef.tlvDataId **shall exist and have a unique value** in the context of respective enclosing ClientServerOperation. ⌋*()*

Please note that [constr_1532] and [constr_1537] do not exclusively apply, there may be ClientServerOperations that have optional arguments as well as non-optional arguments typed a composite data type for which optional elements have been identified.

A scenario like this would obviously be subject to the application of both [constr_1532] and [constr_1537].

4 Diagnostic Mapping

4.1 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 [12] 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
- DiagnosticServiceSwMapping
- DiagnosticEventPortMapping
- DiagnosticOperationCyclePortMapping
- DiagnosticEnableConditionPortMapping
- DiagnosticStorageConditionPortMapping

In order to exemplify the approach, The diagram depicted in Figure 4.1 describes a very simplistic situation where **one** event contained **in one of two** different PPortPrototypes exposed by an AdaptiveApplicationSwComponentType is accessed by the AUTOSAR Adaptive Diagnostic Management on the *AUTOSAR adaptive platform* with the purpose of adding the value to e.g. a DID response telegram.

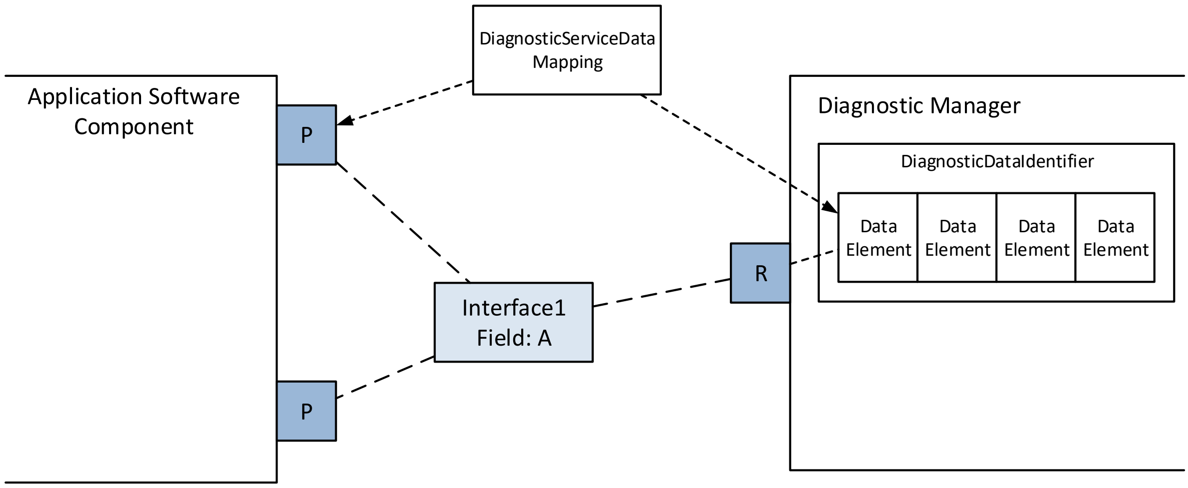

Figure 4.1: Example data exchange for diagnostic purpose

In this situation, the AUTOSAR Adaptive Diagnostic Management obviously needs to be aware which of the two available events has to be accessed in particular.

In other words, the service discovery settings of the AUTOSAR Adaptive Diagnostic Management need to be clear about which of the two available PortPrototypes to connect to.

The process of configuring the AUTOSAR Adaptive Diagnostic Management's service discovery settings accordingly can be assisted by the existence of (in this case) a DiagnosticServiceDataMapping that formally identifies the applicable event in the context of the enclosing PortPrototype.

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 DiagnosticDataElement owned by a DiagnosticDataIdentifier) of the external behavior of the diagnostic stack on the *AUTOSAR adaptive platform*, as described by a corresponding Diagnostic Extract [12].

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, Packageable Element, 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 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 `PortPrototype`s in specific roles to a given use-case.

Although (thanks to the existence of the `ServiceInterface`) the need for involving different `PortPrototype`s in the implementation of a given use case has slightly 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 `PortPrototype`s (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.

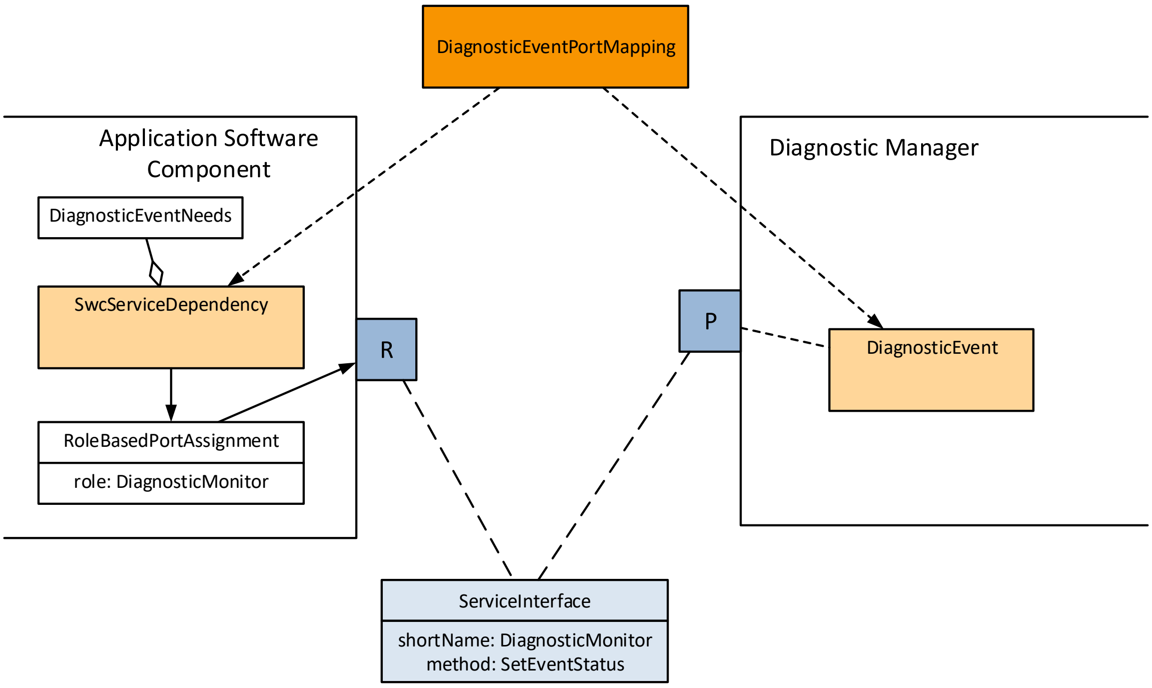

Figure 4.2: Example mapping to associate a `PortPrototype` with a `DiagnosticEvent`

4.2 Diagnostic Data Mapping

[TPS_MANI_01037] 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] 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 `field`s and, at least with respect to the value, `event`s. ⌋*(RS_MANI_00005)*

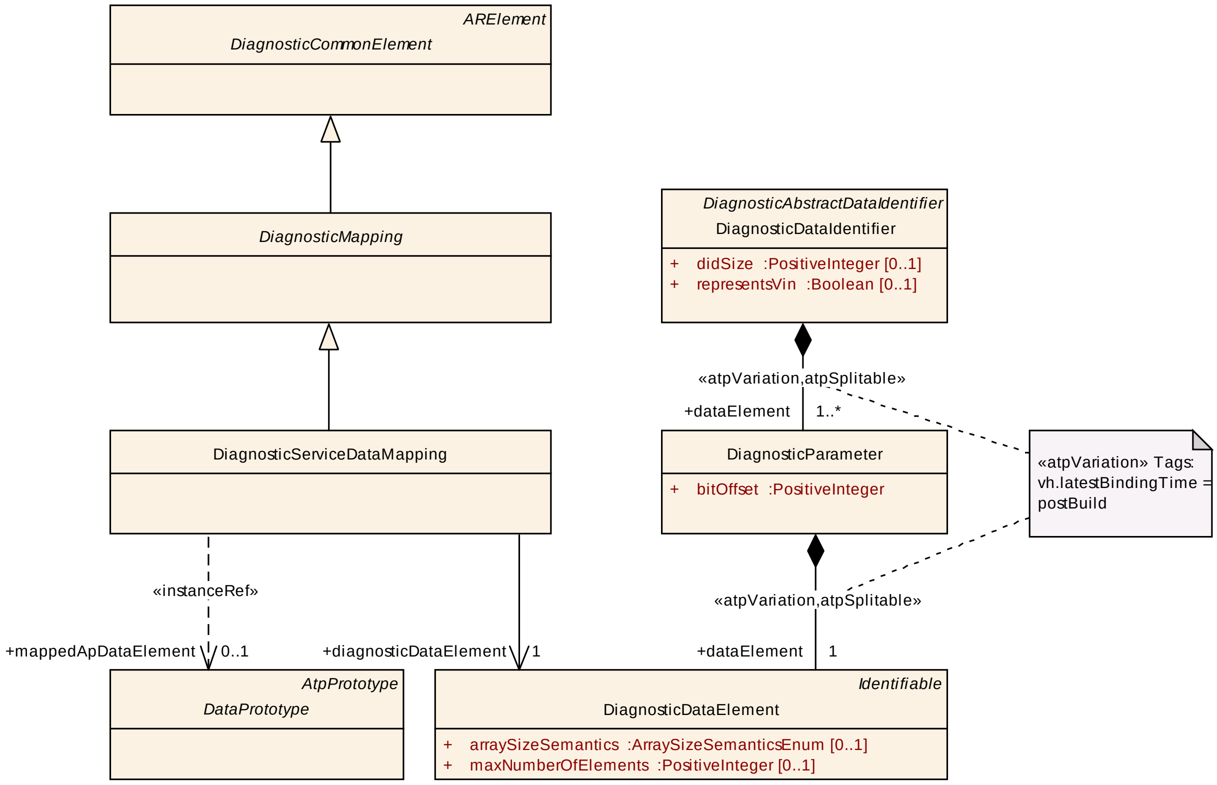

Figure 4.3: Modeling of the diagnostic data mapping

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] 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 not 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. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
diagnosticDataElement	DiagnosticDataElement	1	ref	This represents the applicable payload that corresponds to the referenced DataPrototype in the role mappedDataElement.
mappedApDataElement	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
mappedDataElement	DataPrototype	0..1	iref	This represents the dataElement in the application software that is accessed for diagnostic purpose.

Table 4.2: 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**
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOfElements	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.
swDataDefProps	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 4.3: DiagnosticDataElement

4.3 Diagnostic Software Mapping

[TPS_MANI_01038] Diagnostic software mapping on the *AUTOSAR adaptive platform* ⌈ The diagnostic software mapping on the *AUTOSAR adaptive platform* is created by means of meta-class `DiagnosticServiceSwMapping` that maps a `DiagnosticServiceInstance` to a `SwcServiceDependency` referenced in the role mappedSwcServiceDependencyInExecutable respectively a DiagnosticDataElement in the role diagnosticDataElement. ⌋*(RS_MANI_00005)*

As depicted by Figure 4.4, the application of a DiagnosticServiceSwMapping on the *AUTOSAR adaptive platform* requires the existence of a SwcServiceDependency, defined in the context of an AdaptiveApplicationSwComponentType (see section 3.2).

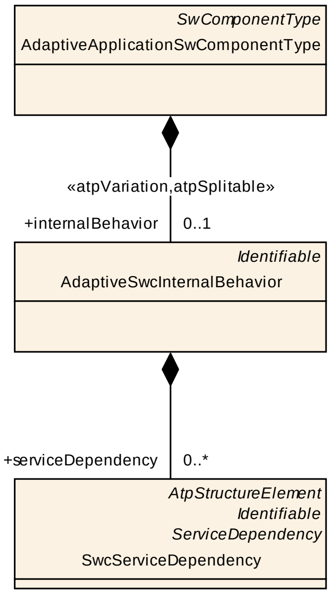

Figure 4.4: Modeling of internal behavior for the modeling of DiagnosticServiceSwMapping

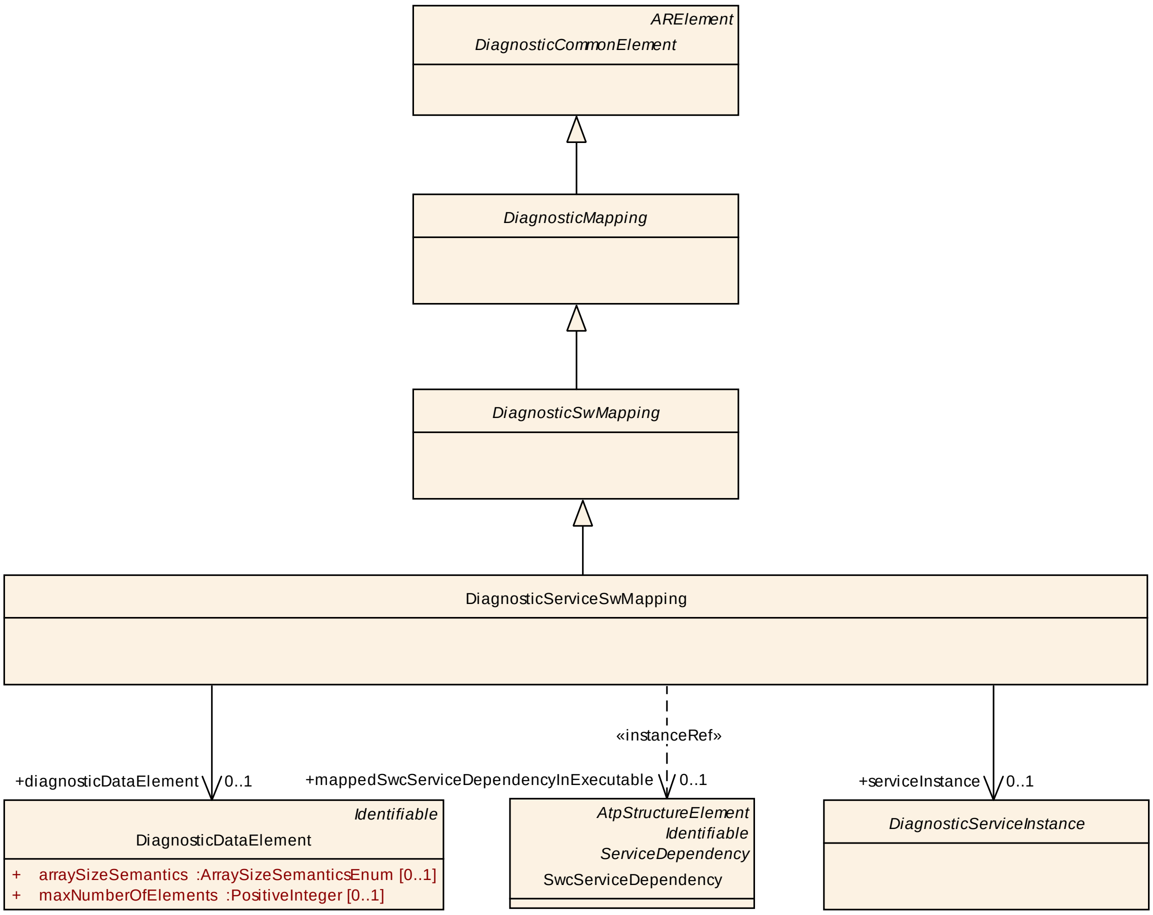

Figure 4.5: Modeling of the diagnostic software mapping

[constr_1499] Target `SwcServiceDependency` of `DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable` ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable` shall **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior`. ⌋ *()*

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 emebdded into the component hiearchy.
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)
serviceInstance	DiagnosticServiceInstance	0..1	ref	This represents the service instance that needs to be considered in this diagnostics service mapping.

Table 4.4: DiagnosticServiceSwMapping

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	RoleBasedDataAssignment	*	aggr	Defines the role of an associated data object of the same component. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
assignedPort	RoleBasedPortAssignment	*	aggr	Defines the role of an associated port of the same component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=assignedPort, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

representedPortGroup	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 PortGroups, a tool can evaluate this information such that all the ports linked via this port group on the same ECU can be found.
serviceNeeds	ServiceNeeds	1	aggr	The associated ServiceNeeds.

Table 4.5: SwcServiceDependency

4.4 Diagnostic Event to Port Mapping

[TPS_MANI_01048] 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)*

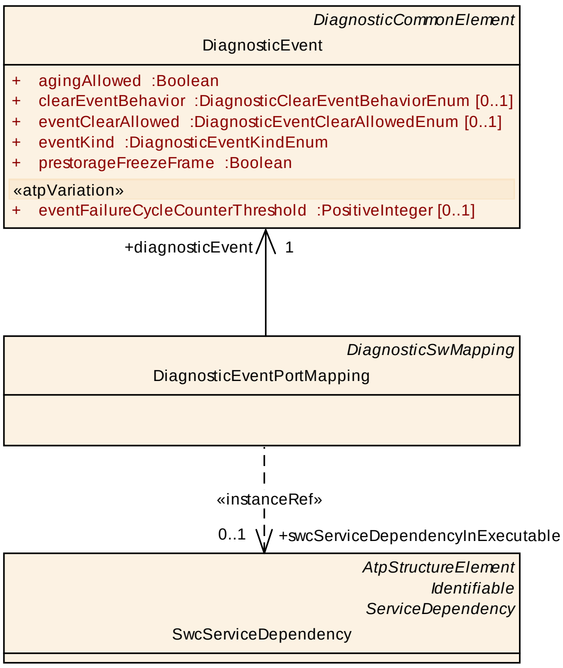

Figure 4.6: Modeling of `DiagnosticEventPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1500] Target `SwcServiceDependency` of `DiagnosticEventPortMapping.swcServiceDependencyInExecutable` ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticEventPortMapping.swcSer-`

viceDependencyInExecutable shall **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior. ⌋()

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***
agingAllowed	Boolean	1	attr	This represents the decision whether aging is allowed for this DiagnosticEvent.
clearEventBehavior	DiagnosticClearEventBehaviorEnum	0..1	attr	This attribute defines the resulting UDS 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
eventKind	DiagnosticEventKindEnum	1	attr	This attribute is used to distinguish between SWC and BSW events.
prestorageFreezeFrame	Boolean	1	attr	This attribute describes whether the Prestorage of FreezeFrames is supported by the assigned event or not. True: Prestorage of FreezeFrames is supported False: Prestorage of FreezeFrames is not supported

Table 4.6: 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.
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 4.7: DiagnosticEventPortMapping

4.5 Diagnostic Operation Cycle to Port Mapping

[TPS_MANI_01049] 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 `PortPrototype`s is created by using the `DiagnosticEventPortMapping` that refers to a `DiagnosticOperationCycle` in the role `operationCycle` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋ *(RS_MANI_00005)*

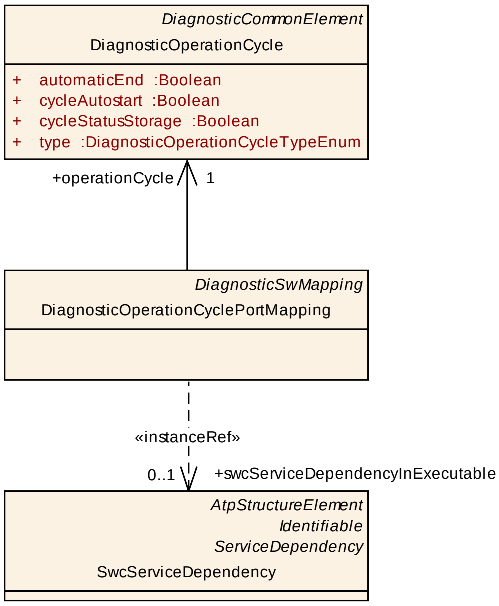

Figure 4.7: Modeling of `DiagnosticOperationCyclePortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1501] 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`. ⌋*()*

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, MultilanguageReferrable, 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.
cycleStatusStorage	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	DiagnosticOperationCycleTypeEnum	1	attr	Operation cycles types for the Dem to be supported by cycle-state APIs.

Table 4.8: 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	DiagnosticOperationCycle	1	ref	Reference to the DiagnosticOperationCycle that is assigned to SWC service ports with DiagnosticOperationCycleNeeds.
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 DiagnosticOperationCyclePortMapping 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 4.9: DiagnosticOperationCyclePortMapping

4.6 Diagnostic Enable Condition to Port Mapping

[TPS_MANI_01050] 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 `DiagnosticEventPortMapping` that refers to a `DiagnosticEnableCondition` in the role `enableCondition` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋ *(RS_MANI_00005)*

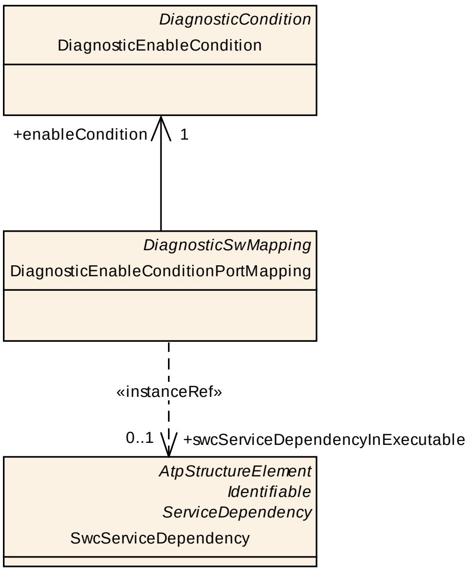

Figure 4.8: Modeling of DiagnosticEnableConditionPortMapping for the usage on the *AUTOSAR adaptive platform*

[constr_1502] 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. ⌋ *()*

Class	**DiagnosticEnableCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of an enable condition. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Diagnostic Condition, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 4.10: DiagnosticEnableCondition

Class	**DiagnosticEnableConditionPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticEnableConditionNeeds the DiagnosticEnableCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Diagnostic Mapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, Packageable Element, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

enableCondition	DiagnosticEnableCondition	1	ref	Reference to the EnableCondition which is mapped to a SWC service port with DiagnosticEnableConditionNeeds.
swcFlatServiceDependency	SwcServiceDependency	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.
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This aggregation allows for the usage of the DiagnosticEnableConditionPortMapping 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 4.11: DiagnosticEnableConditionPortMapping

4.7 Diagnostic Storage Condition to Port Mapping

[TPS_MANI_01051] 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 `DiagnosticEventPortMapping` that refers to a `DiagnosticStorageCondition` in the role `diagnosticStorageCondition` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋*(RS_MANI_00005)*

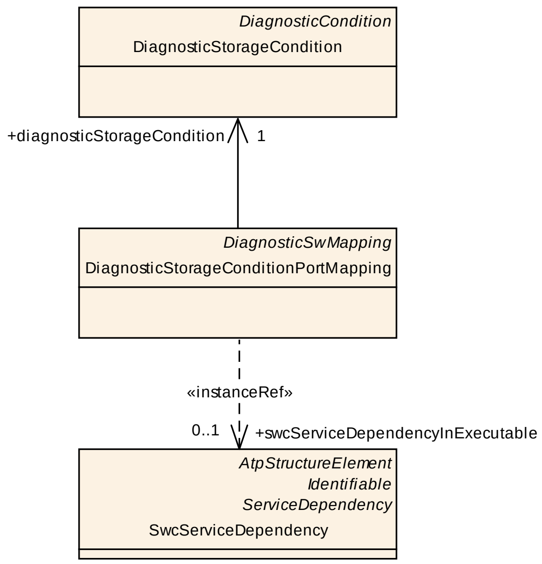

Figure 4.9: Modeling of `DiagnosticStorageConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1503] 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. ⌋*()*

Class	**DiagnosticStorageCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of a storage condition. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Diagnostic Condition, 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 DiagnosticStorageCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Diagnostic Mapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, Packageable Element, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
diagnostic StorageCo ndition	DiagnosticStora geCondition	1	ref	Reference to the StorageCondition which is mapped to a SWC service port with DiagnosticStorageConditionNeeds.
swcFlatSer viceDepen dency	SwcServiceDep endency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcServic eDepende ncyInExec utable	SwcServiceDep endency	0..1	iref	This aggregation allows for the usage of the DiagnosticStorageConditionPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft
swcServic eDepende ncyInSyste m	SwcServiceDep endency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.13: DiagnosticStorageConditionPortMapping

5 REST Design

5.1 Overview

Important note: the AUTOSAR SWS REST [13] 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**" [14] (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] 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 5.1.

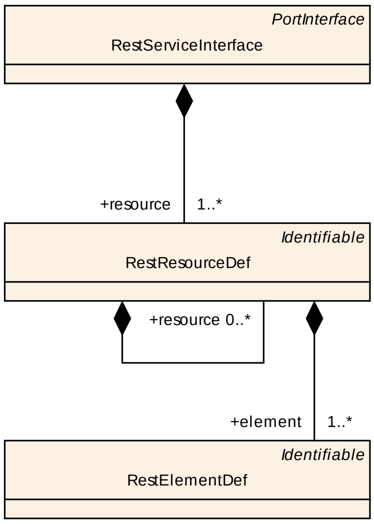

Figure 5.1: Big picture to REST modeling

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 16.

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.

5.2 REST Service Interface

As depicted in Figure 5.2, 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.

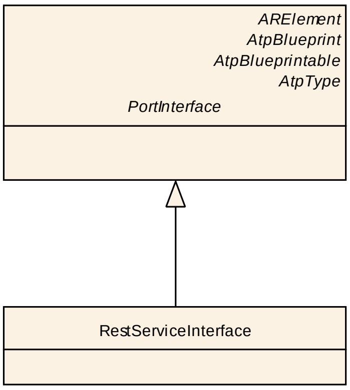

Figure 5.2: Modeling of the REST service

[TPS_MANI_01105] 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::ApplicationDesign::PortInterface::REST			
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	1..*	aggr	This aggregation represents the collectioin of resources owned by the enclosing REST service. **Tags:** atp.Status=draft

Table 5.1: RestServiceInterface

5.3 REST Resource

[TPS_MANI_01120] Recursive definition of RestResourceDef ⌈ The definition of RestResourceDef supports the aggregation of other RestResourceDef. In other words, it is possible to created a nested definition of RestResourceDefs. ⌋ *(RS_MANI_00033)*

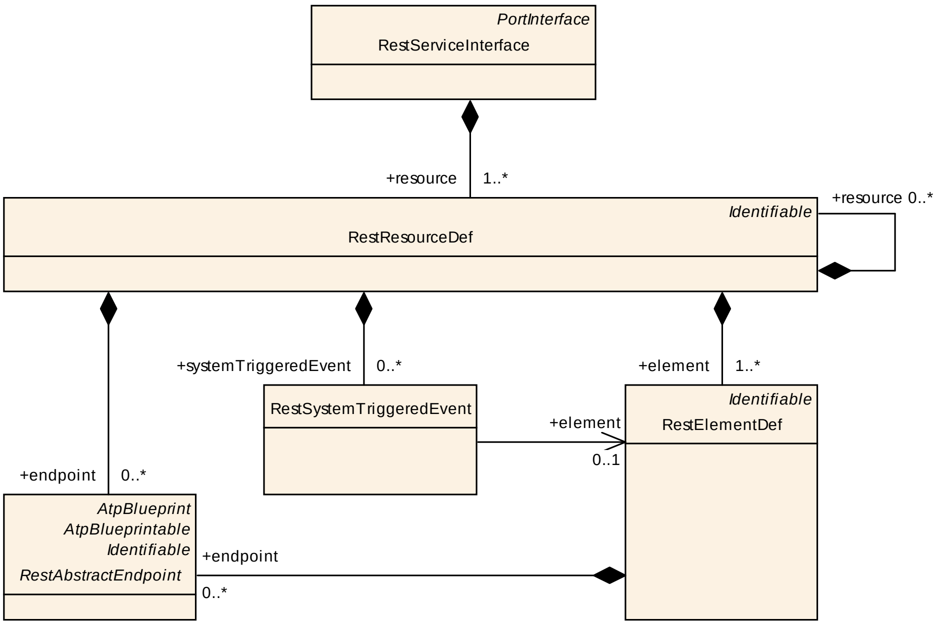

Figure 5.3: Modeling of the REST resource level

Class	**RestResourceDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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
systemTriggeredEvent	RestSystemTriggeredEvent	*	aggr	This represents the collection of system triggered events for the enclosing resource. **Tags:** atp.Status=draft

Table 5.2: RestResourceDef

[TPS_MANI_01121] 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)*

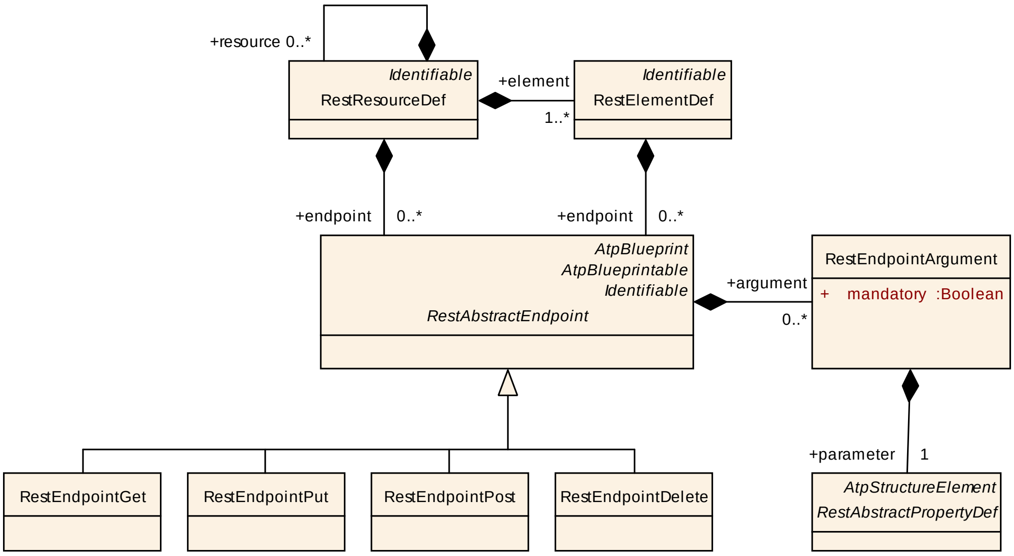

Figure 5.4: Modeling of the REST endpoints

[TPS_MANI_01122] 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::ApplicationDesign::PortInterface::REST			
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			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

argument	RestEndpointArgument	*	aggr	Some endpoints can require a list of arguments. **Tags:** atp.Status=draft

Table 5.3: RestAbstractEndpoint

Class	**RestEndpointPut**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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, RestAbstractEndpoint			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 5.4: RestEndpointPut

Class	**RestEndpointGet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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, RestAbstractEndpoint			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 5.5: RestEndpointGet

Class	**RestEndpointPost**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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, RestAbstractEndpoint			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 5.6: RestEndpointPost

Class	RestEndpointDelete			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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, RestAbstractEndpoint			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 5.7: RestEndpointDelete

Class	RestEndpointArgument			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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 5.8: RestEndpointArgument

[TPS_MANI_01123] 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)*

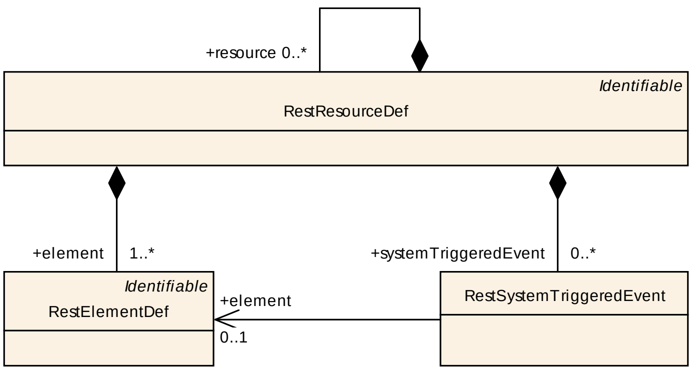

Figure 5.5: Modeling of the REST system triggered event

Class	RestSystemTriggeredEvent			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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**
element	RestElementDef	0..1	ref	This reference represent the element that is linked to the system triggered event. **Tags:** atp.Status=draft

Table 5.9: RestSystemTriggeredEvent

5.4 REST Element

[TPS_MANI_01124] 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::ApplicationDesign::PortInterface::REST			
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 5.10: RestElementDef

Class	RestAbstractPropertyDef (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 5.11: RestAbstractPropertyDef

As depicted by Figure 5.6, 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 an `ImplementationDataType`.

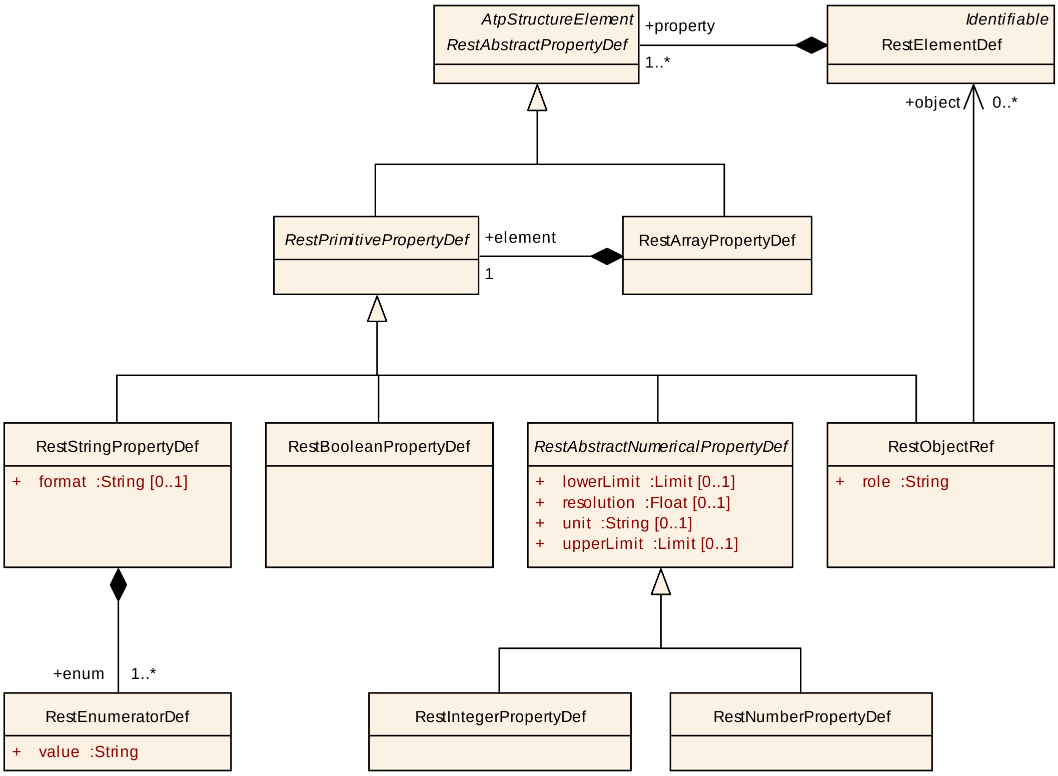

Figure 5.6: Modeling of the REST elements

[TPS_MANI_01125] 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::ApplicationDesign::PortInterface::REST			
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			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 5.12: RestPrimitivePropertyDef

Class	RestArrayPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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	RestPrimitivePropertyDef	1	aggr	This aggregation represents the definition of the base element type of the array property **Tags:** atp.Status=draft

Table 5.13: RestArrayPropertyDef

Class	RestBooleanPropertyDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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 5.14: RestBooleanPropertyDef

[TPS_MANI_01126] 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::ApplicationDesign::PortInterface::REST			
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, Multilanguage Referrable, 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 5.15: RestStringPropertyDef

Class	RestEnumeratorDef			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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 5.16: RestEnumeratorDef

[TPS_MANI_01127] 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] 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::ApplicationDesign::PortInterface::REST			
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, Multilanguage Referrable, Referrable, RestAbstractPropertyDef, RestPrimitivePropertyDef			
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

[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.

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 5.17: RestAbstractNumericalPropertyDef

Class	**RestIntegerPropertyDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
Note	This meta-class represents the ability to define a REST property with an integer semantics. **Tags:** atp.Status=draft			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, RestAbstractNumericalPropertyDef, RestAbstractPropertyDef, RestPrimitivePropertyDef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 5.18: RestIntegerPropertyDef

Class	**RestNumberPropertyDef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::REST			
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 5.19: RestNumberPropertyDef

[TPS_MANI_01129] `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::ApplicationDesign::PortInterface::REST			
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, Multilanguage Referrable, 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
role	String	1	attr	This attribute represents the ability to define a role for the reference to another element. **Tags:** atp.Status=draft

Table 5.20: RestObjectRef

The application of the attribute RestObjectRef.role is sketched in Figure 5.7. 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.

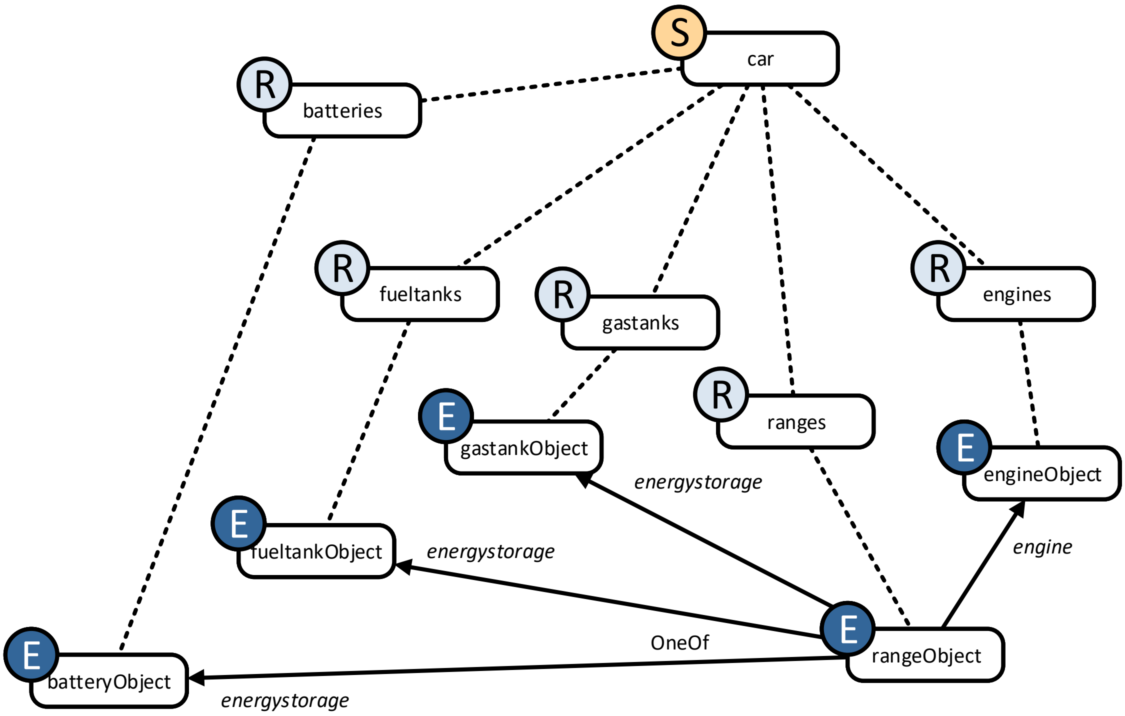

Figure 5.7: Example of the usage of the `role` attribute

6 Application Manifest

6.1 Overview

The purpose of the application 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 7.

The top-level element of the Application 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] 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)*

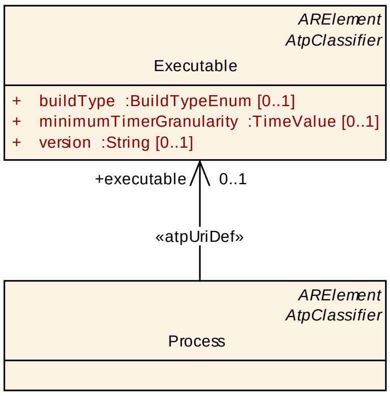

Figure 6.1: Relation of meta-classes `Executable` and `Process`

Class	**Process**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process			
Note	This meta-class provides information required to execute the referenced executable. **Tags:** atp.Status=draft; atp.recommendedPackage=Processes			
Base	ARElement, ARObject, AtpClassifier, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
applicationModeMachine	ModeDeclarationGroupPrototype	0..1	aggr	Set of ApplicationStates (Modes) that are defined for the process. **Tags:** atp.Status=draft
executable	Executable	0..1	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
modeDependentStartupConfig	ModeDependentStartupConfig	*	aggr	Applicable startup configurations. **Tags:** atp.Status=draft

Table 6.1: Process

Please note that the meta-model, as depicted in Figure 6.1 supports the existence of two or more `Process`es 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 6.2

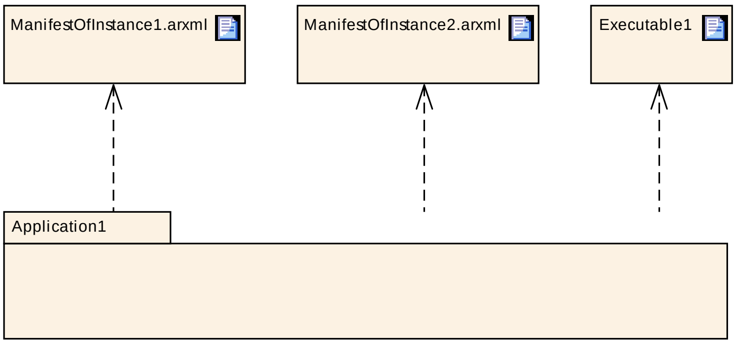

Figure 6.2: Example deployment where one Executable is bundled with two ARXML files that each contain the description of one Process

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 6.2.

The supported application states that are defined in the Process.applicationModeMachine are described in more detail in [15] by [SWS_EM_01052], [SWS_EM_01053], [SWS_EM_01055].

6.2 Startup Configuration

The configuration of startup behavior is an essential part of the application manifest.

[TPS_MANI_01012] Formal modeling of application startup behavior ⌈ The formal modeling of application startup behavior is implemented by means of the aggregation of meta-class ModeDependentStartupConfig in the role Process.modeDependentStartupConfig. ⌋*(RS_MANI_00007)*

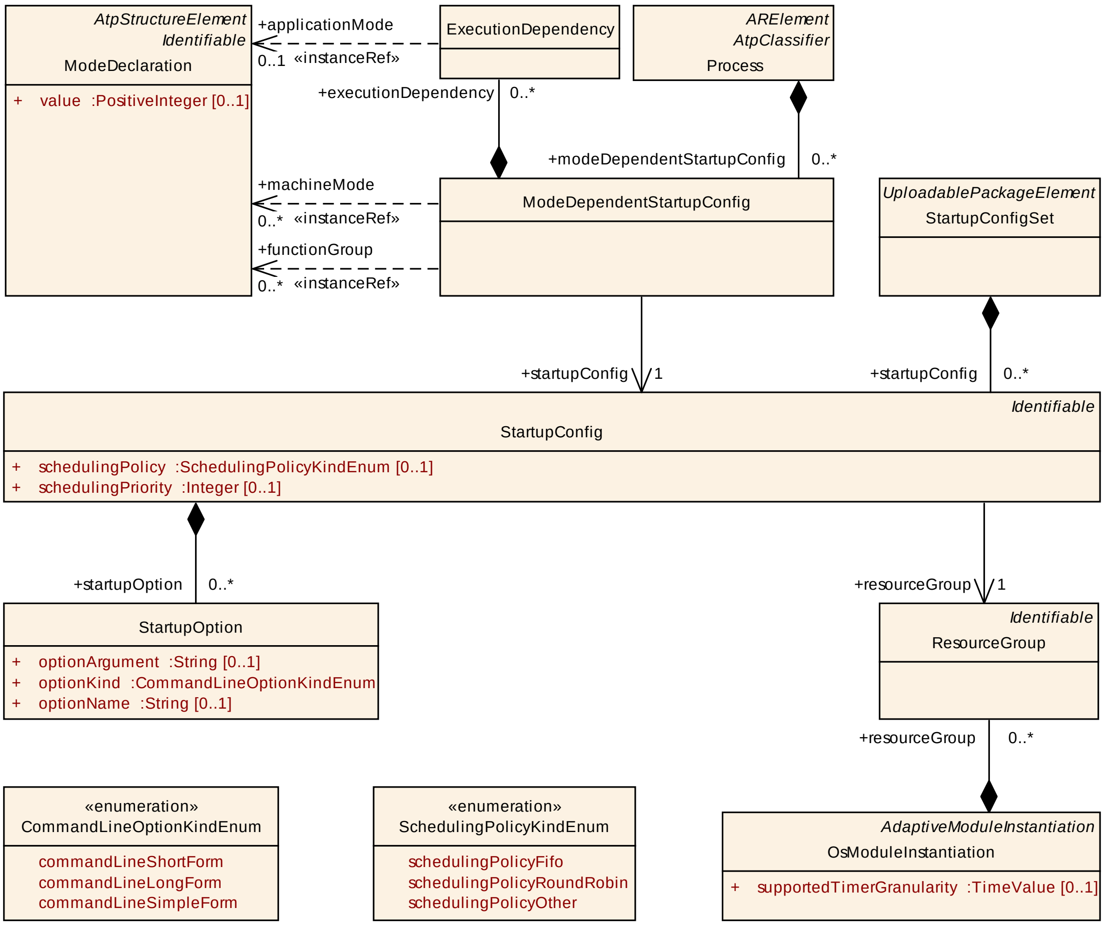

Figure 6.3: Content of a `Process`

6.2.1 Mode-dependent Startup Configuration

As a consequence of the reference from the `ModeDependentStartupConfig` to `ModeDeclaration` the `Application Manifest` is defined for a specific `Machine` to which the binary and the Manifest is deployed.

[TPS_MANI_01045] `Process.modeDependentStartupConfig` that does not refer to a `ModeDeclaration` ⌈ If one `Process.modeDependentStartupConfig` does not refer to a `ModeDeclaration` then this means that one approach to execute the `Process` does not depend on any `ModeDeclaration`. ⌋*(RS_MANI_00007)*

It is necessary to specify constraints [constr_1504] and [constr_3396] to regulate the number of `ModeDependentStartupConfig` that refer to the same `ModeDeclaration` in the context of one `Process` because the resulting startup configuration would be ambiguous.

[constr_1504] Number of `Process.modeDependentStartupConfig` that refer to the same `machineMode` ⌈ Within the context of a given `Process`, no two `modeDependentStartupConfig` shall refer to the same `ModeDeclaration` in the role `machineMode`. ⌋*()*

[constr_3396] Number of Process.modeDependentStartupConfig that refer to the same functionGroup ⌈ Within the context of a given Process, no two modeDependentStartupConfig shall refer to the same ModeDeclaration in the role functionGroup. ⌋*()*

In the same spirit, it is necessary to limit (see [constr_1505]) the number of ModeDependentStartupConfig if there is one ModeDependentStartupConfig that does not refer to any ModeDeclaration in the context of one Process.

That is, the existence of multiple modeDependentStartupConfigs (within the context of one Process) with no reference to a ModeDeclaration would also create an ambiguous startup configuration.

[constr_1505] Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration ⌈ If a Process has **one** modeDependentStartupConfig that does not refer to a ModeDeclaration then the Process shall not aggregate **any other** modeDependentStartupConfig. ⌋*()*

[constr_3397] ModeDependentStartupConfig that refers to a functionGroup and to a machineMode ⌈ If a Process has **one** modeDependentStartupConfig that does refer to a functionGroup and to a machineMode then the Process shall not aggregate **any other** modeDependentStartupConfig. ⌋*()*

[constr_3398] ModeDependentStartupConfig that refers to function group modes of different function groups ⌈ If a Process has **one** modeDependentStartupConfig that refers to ModeDeclarations of different ModeDeclarationGroups in the role functionGroup then the Process shall not aggregate **any other** modeDependentStartupConfig. ⌋*()*

[TPS_MANI_01046] Semantics of ModeDependentStartupConfig.machineMode ⌈ The ModeDeclarations referenced in the role ModeDependentStartupConfig.machineMode shall be considered in a way such that the ModeDependentStartupConfig applies if **any** of the referenced ModeDeclarations is active.

In other words, the ModeDeclarations are or-ed for the determination of whether a ModeDependentStartupConfig is applicable. ⌋*(RS_MANI_00007)*

[TPS_MANI_03153] Semantics of ModeDependentStartupConfig.functionGroup ⌈ The ModeDeclarations referenced in the role ModeDependentStartupConfig.functionGroup shall be considered in a way such that the ModeDependentStartupConfig applies if **any** of the referenced ModeDeclarations is active.

In other words, the ModeDeclarations are or-ed for the determination of whether a ModeDependentStartupConfig is applicable. ⌋*(RS_MANI_00007)*

Class	**ModeDependentStartupConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process			
Note	This meta-class defines the startup configuration for the process depending on a collection of machine states. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
executionDependency	ExecutionDependency	*	aggr	This attribute defines that all processes that are referenced via the ExecutionDependency shall be launched and shall reach a certain ApplicationState before the referencing process is started. **Tags:** atp.Status=draft
functionGroup	ModeDeclaration	*	iref	This represent the applicable functionGroup. **Tags:** atp.Status=draft
machineMode	ModeDeclaration	*	iref	This represent the applicable machineMode. **Tags:** atp.Status=draft
startupConfig	StartupConfig	1	ref	Reference to a reusable startup configuration with startup parameters. **Tags:** atp.Status=draft

Table 6.2: ModeDependentStartupConfig

Class	**ModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this ModeDeclaration.

Table 6.3: ModeDeclaration

[TPS_MANI_01013] Semantics of meta-class `ModeDependentStartupConfig` ⌈
The purpose of meta-class `ModeDependentStartupConfig` to qualify the startup configuration represented by meta-class `StartupConfig` for specific `ModeDeclaration`s.

In other words, the intention is to express that the `StartupConfig` is applicable if the mode machines that control the startup are in the modes represented by the `ModeDeclaration` referenced in the role `ModeDependentStartupConfig.machineMode` and/or `ModeDependentStartupConfig.functionGroup`. ⌋*(RS_MANI_00007)*

Please note that the corresponding SWS for the definition of the Execution Manager may refer to *states*. Similar to the situation on the *classic AUTOSAR platform*, the term *mode* used in this document directly corresponds to a *state* on the level of middleware software.

Class	**StartupConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process			
Note	This meta-class represents a reusable startup configuration for processes.. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
resourceGroup	ResourceGroup	1	ref	Reference to applicable resource groups. **Tags:** atp.Status=draft
schedulingPolicy	SchedulingPolicyKindEnum	0..1	attr	This attribute represents the ability to define the scheduling policy for the initial thread of the application.
schedulingPriority	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 6.4: StartupConfig

6.2.2 Scheduling

[TPS_MANI_01061] 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)*

Enumeration	**SchedulingPolicyKindEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process
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**
schedulingPolicyFifo	This attribute represents the setting for a FIFO scheduling policy. **Tags:** atp.EnumerationValue=0

scheduling PolicyOther	This attribute represents the setting for a custom scheduling policy. **Tags:** atp.EnumerationValue=2
scheduling PolicyRound Robin	This attribute represents the setting for a round robin scheduling policy **Tags:** atp.EnumerationValue=1

Table 6.5: SchedulingPolicyKindEnum

6.2.3 Startup Options

[TPS_MANI_01014] 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::Deployment::Process			
Note	Collection of reusable startup configurations for processes. **Tags:** atp.Status=draft; atp.recommendedPackage=StartupConfigSets			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
startupCon fig	StartupConfig	*	aggr	Startup configuration that is contained in the StartupConfigSet **Tags:** atp.Status=draft

Table 6.6: 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] 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] 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] 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] 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::Deployment::Process			
Note	This meta-class represents a single startup option consisting of option name and an optional argument. **Tags:** 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 6.7: StartupOption

Enumeration	CommandLineOptionKindEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process
Note	This enum defines the different styles how the command line option appear 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 6.8: CommandLineOptionKindEnum

6.2.4 Resources

Meta-class StartupConfig also supports the specification of a relation to a resource group.

[TPS_MANI_01017] Relation of startup configuration to resource group ⌈ The modeling of a resource group is possible by means of meta-class ResourceGroup and the association from StartupConfig to ResourceGroup in the role resource-Group ⌋*(RS_MANI_00007)*

Class	**ResourceGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::AdaptiveModule Implementation			
Note	This meta-class represents a resource group. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 6.9: ResourceGroup

6.2.5 Execution Dependency

[TPS_MANI_01041] Startup configuration supports the definition of a launch dependency ⌈ The modeling of startup configuration also supports the definition of a launch dependency, formalized by the meta-class ExecutionDependency that is aggregated by ModeDependentStartupConfig in the role executionDependency.

The ExecutionDependency allows to define a dependency to a process that needs to be in a specific application state before the process that aggregates the ExecutionDependency via ModeDependentStartupConfig is launched. ⌋*(RS_MANI_00007)*

Please note that, in addition to the explicit definition a launch dependency, there are further ways to specify a dependency between different applications. For example there is an implicit dependency between an application that offers a given service and an application that requires this service.

Obviously, the most elegant approach for startup in this case would be to launch the server application first and then launch the client application.

Service discovery would still work if this implicit dependency is not observed but the inverse launch order may lead to a certain delay until the connection between the server and the client is fully set up.

Small delays may add up and create a significant offset to the overall startup time of an ECU running the *AUTOSAR adaptive platform*. Therefore, it may be advised to observe the implicit launch dependency between applications based on the configuration of service-oriented communication.

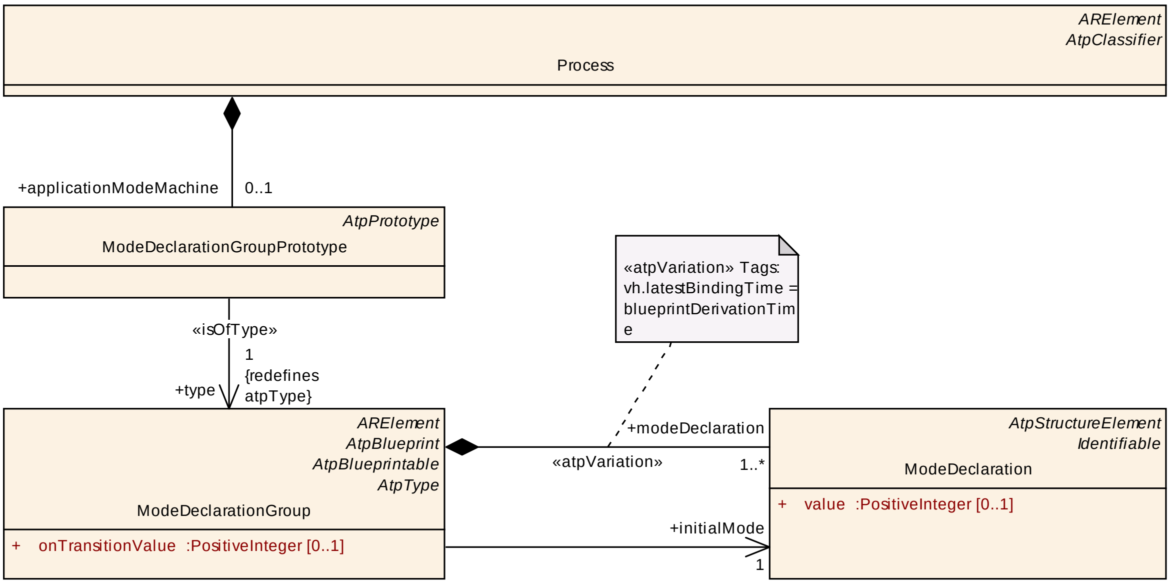

Figure 6.4: Modeling of how `Process` relates to `ModeDeclaration`

Class	**ExecutionDependency**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process			
Note	This element defines an ApplicationState in which a dependent process needs to be before the process that aggregates the ExecutionDependency element can be started. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
application Mode	ModeDeclaration	0..1	iref	This represent the applicable modeDeclaration that represents an ApplicationState. **Tags:** atp.Status=draft

Table 6.10: ExecutionDependency

However, it may become counterproductive if – in addition to the existence of implicit dependencies – further explicit dependencies are created by means of using the `ModeDependentStartupConfig.executionDependency`.

This may very easily lead to contradictions that could not be resolved conflict-free and may lead to increased startup times.

[constr_1484] Applicability of `ModeDependentStartupConfig.executionDependency` ⌈ The following restrictions apply for the existence of `ModeDependentStartupConfig.executionDependency`:

- The `Process` that contains the `applicationMode` that is referenced by the `ExecutionDependency` shall **only** reference an `Executable` that in turn is referenced by an AdaptiveAutosarApplication that has the value of attribute category set to PLATFORM_LEVEL (see [TPS_MANI_01009]).

- The Process that aggregates the ExecutionDependency via ModeDependentStartupConfig that refers indirectly to another Process via the applicationMode shall **only** reference an Executable that in turn is referenced by an AdaptiveAutosarApplication that has the value of attribute category set to PLATFORM_LEVEL.

⌋*()*

In other words: the explicit launch dependency is reserved for platform modules that, in all likelihood, do not use service-oriented communication to communicate with each other.

[constr_3350] Consistent value of category for AdaptiveAutosarApplications referencing an Executable ⌈ All AdaptiveAutosarApplications that reference a specific Executable shall have the value of attribute category set to the same value. ⌋*()*

6.2.6 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. TheModeDependentStartupConfig 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 8.5 in more detail. The usage of Function Groups is described in more detail in [15].

[TPS_MANI_03152] Assignment of a ModeDependentStartupConfig to a function group state ⌈ The ModeDependentStartupConfig is assigned to a function group state with the functionGroup reference. ⌋*()*

7 Service Instance Manifest

7.1 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.

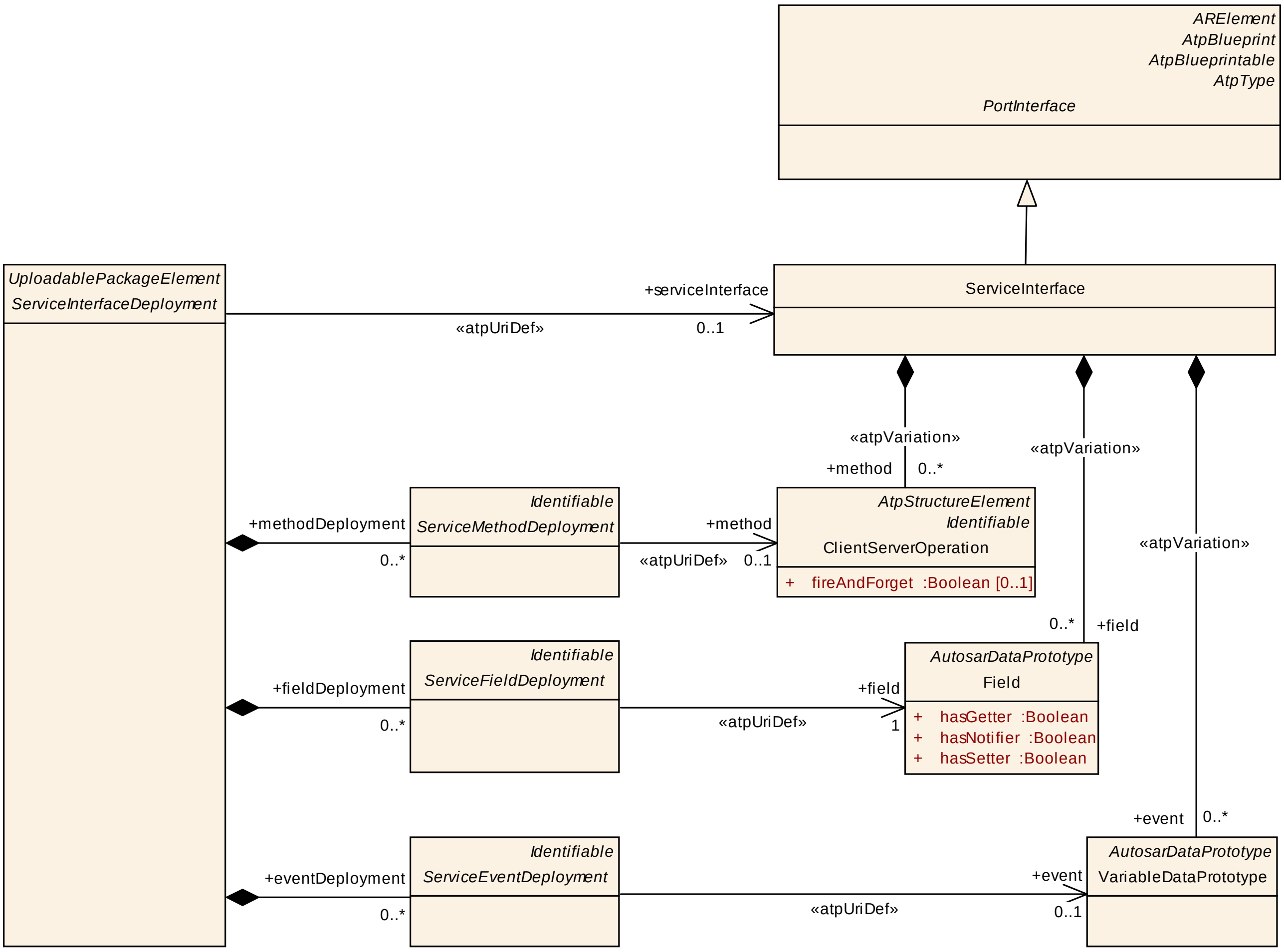

Figure 7.1: Deployment-related modeling of ServiceInterface

[TPS_MANI_03036] 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::Deployment::ServiceInterfaceDeployment			
Note	Middleware transport layer specific configuration settings for the ServiceInterface and all contained ServiceInterface elements. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
eventDeployment	ServiceEventDeployment	*	aggr	Middleware transport layer specific configuration settings for an Event that is defined in the ServiceInterface. **Tags:** atp.Status=draft
fieldDeployment	ServiceFieldDeployment	*	aggr	Middleware transport layer specific configuration settings for a Field that is defined in the ServiceInterface. **Tags:** atp.Status=draft
methodDeployment	ServiceMethodDeployment	*	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 7.1: ServiceInterfaceDeployment

[TPS_MANI_03037] 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] 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] 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] 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] 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] 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::Deployment::ServiceInterface Deployment			
Note	This abstract meta-class represents the ability to specify a deployment of a Method to a middleware transport layer. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
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 7.2: ServiceMethodDeployment

Class	**ServiceEventDeployment (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	This abstract meta-class represents the ability to specify a deployment of an Event to a middleware transport layer. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
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 7.3: ServiceEventDeployment

Class	**ServiceFieldDeployment (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	This abstract meta-class represents the ability to specify a deployment of a Field to a middleware transport layer. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
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 7.4: ServiceFieldDeployment

7.1.1 SOME/IP Service Interface Deployment

This chapter describes the SOME/IP deployment of a `ServiceInterface`.

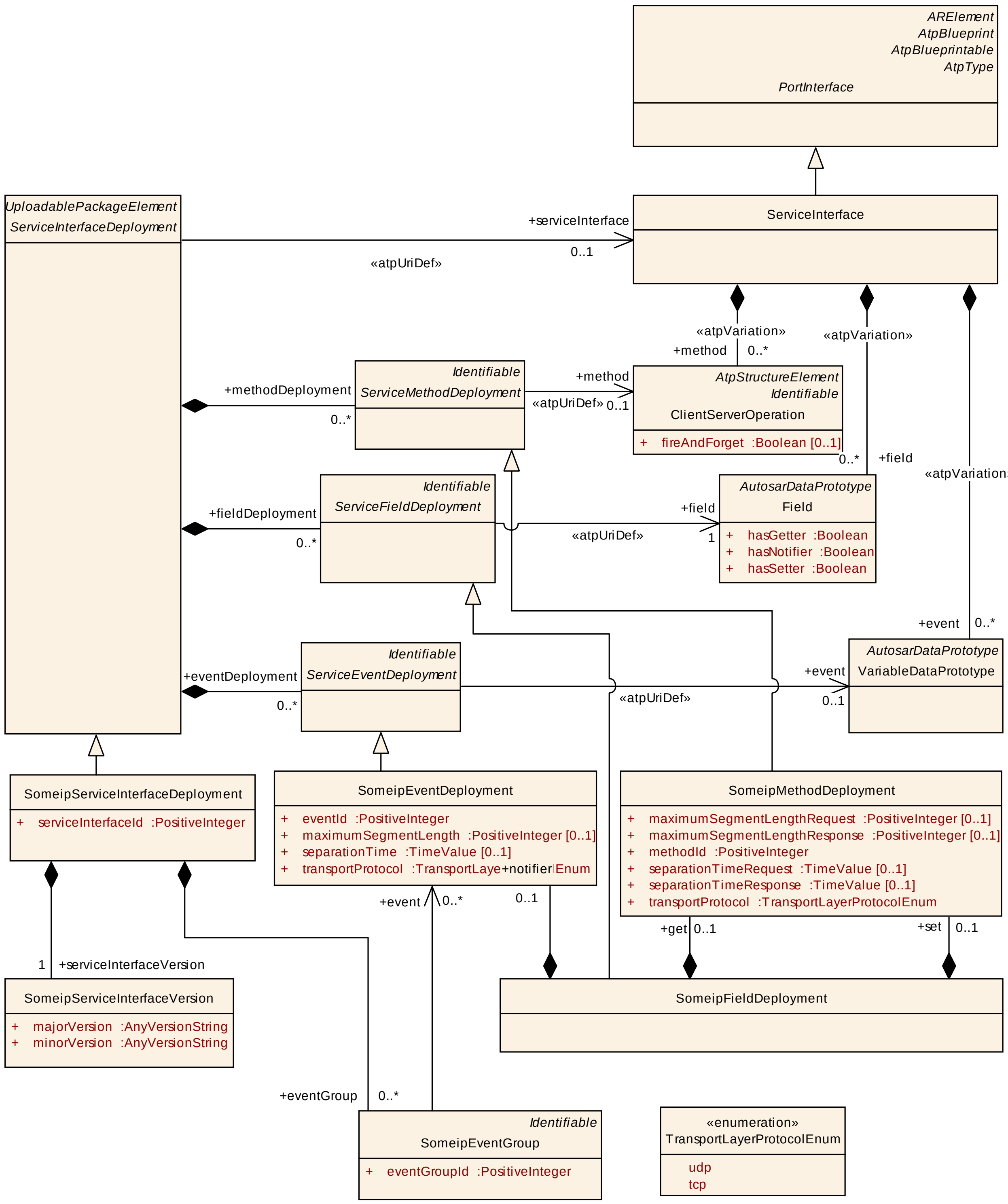

Figure 7.2: SOME/IP deployment of ServiceInterface

[TPS_MANI_03040] 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)*

[TPS_MANI_03041] 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] 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] 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)*

[constr_3303] ANY not allowed for SomeipServiceInterfaceDeployment.serviceInterfaceVersion ⌈ The value ANY is not allowed for the majorVersion and minorVersion of the SomeipServiceInterfaceDeployment.serviceInterfaceVersion. ⌋*()*

Class	**SomeipServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a ServiceInterface. **Tags:** 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.
serviceInterfaceVersion	SomeipServiceInterfaceVersion	1	aggr	The SOME/IP major and minor Version of the Service. **Tags:** atp.Status=draft

Table 7.5: SomeipServiceInterfaceDeployment

Class	SomeipServiceInterfaceVersion			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
Note	This meta-class represents the ability to describe a version of a SOME/IP ServiceInterface. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	Type	Mul.	Kind	Note
majorVersion	AnyVersionString	1	attr	Major Version of the ServiceInterface. Value can be set to a number that represents the Major Version of the searched service or to ANY.
minorVersion	AnyVersionString	1	attr	Minor Version of the ServiceInterface. Value can be set to a number that represents the Minor Version of the searched service or to ANY.

Table 7.6: SomeipServiceInterfaceVersion

Class	SomeipEventGroup			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterfaceDeployment			
Note	Grouping of events and notification events inside a ServiceInterface in order to allow subscriptions. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Type	Mul.	Kind	Note
event	SomeipEventDeployment	*	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 7.7: SomeipEventGroup

[TPS_MANI_03043] 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] 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`. ⌋*()*

[TPS_MANI_03050] 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] 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 Machine with a SomeipServiceInstanceToMachineMapping with a configured udpPort. ⌋*()*

[constr_3308] 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 Machine with a SomeipServiceInstanceToMachineMapping with a configured tcpPort. ⌋*()*

[TPS_MANI_03067] 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] 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::Deployment::ServiceInterface Deployment			
Note	SOME/IP configuration settings for an Event. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceEvent Deployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
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.

maximumSegmentLength	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.
transportProtocol	TransportLayerProtocolEnum	1	attr	This attribute defines over which Transport Layer Protocol this event is intended to be sent.

Table 7.8: SomeipEventDeployment

[TPS_MANI_03044] 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] 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`. ⌋*()*

[TPS_MANI_03051] Usage of `SomeipMethodDeployment.transportProtocol` ⌈ The value of `SomeipMethodDeployment.transportProtocol` defines over which Transport Layer Protocol this method is provided. ⌋*(RS_MANI_00024)*

[constr_3309] `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 `Machine` with a `SomeipServiceInstanceToMachineMapping` with a configured `udpPort`. ⌋*()*

[constr_3310] `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 `Machine` with a `SomeipServiceInstanceToMachineMapping` with a configured `tcpPort`. ⌋*()*

[TPS_MANI_03068] 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] 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] 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::Deployment::ServiceInterface Deployment			
Note	SOME/IP configuration settings for a Method. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceMethod Deployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
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 7.9: SomeipMethodDeployment

Class	**SomeipServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::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.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. SOME/IP Service Discovery: Send in the SD-SubscribeEventGroupAck Message to client (answer to SD-SubscribeEventGroup). Event: This is the destination-port where the server sends the multicast event messages if the mulicastThreshold of the corresponding ProvidedEventGroupInSomeipServiceInstance is exceeded.
ipv4MulticastIpAddress	Ip4AddressString	0..1	attr	Multicast IPv4 Address that is transmitted in the EventGroupSubscribeAck message for all available EventGroups that are available in the ProvidedSomeipServiceInstance.
ipv6MulticastIpAddress	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. 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.
udpPort	PositiveInteger	0..1	attr	UdpPort configuration that is used for Method and Event communication in IP-Unicast case. 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 7.10: SomeipServiceInstanceToMachineMapping

[TPS_MANI_03057] 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::Deployment::ServiceInterface Deployment			
Note	SOME/IP configuration settings for a Field. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceFieldDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
get	SomeipMethod Deployment	0..1	aggr	This aggregation represents the setting of the get method. **Tags:** atp.Status=draft
notifier	SomeipEventDe ployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:** atp.Status=draft
set	SomeipMethod Deployment	0..1	aggr	This aggregation represents the settings of the set method **Tags:** atp.Status=draft

Table 7.11: SomeipFieldDeployment

[constr_3362] 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] 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. ⌋*()*

7.1.2 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.

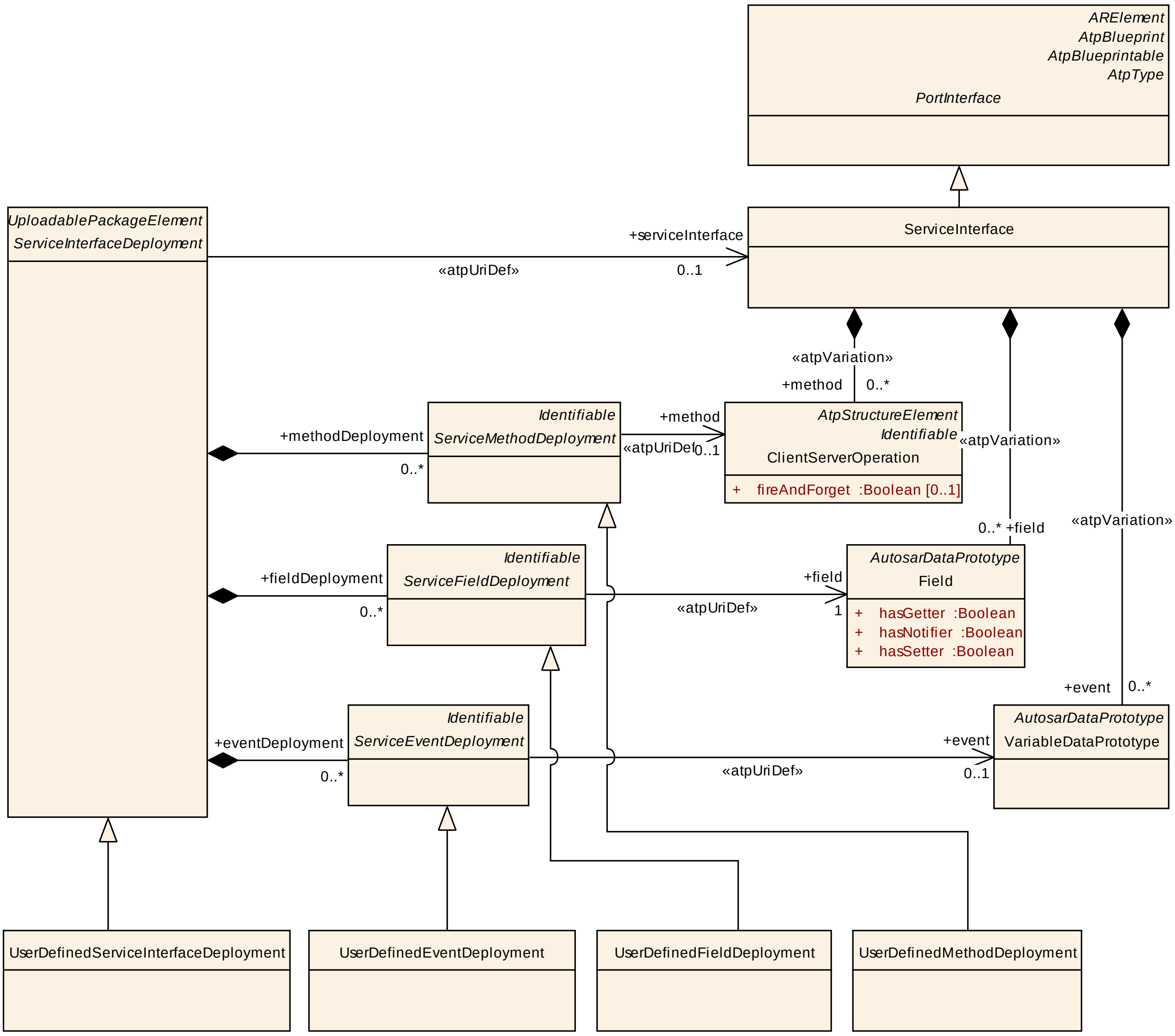

Figure 7.3: User defined deployment of ServiceInterface

[TPS_MANI_03045] 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 is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	UserDefined configuration settings for a ServiceInterface. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInterfaceDeployments			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, ServiceInterfaceDeployment, UploadablePackage Element			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 7.12: UserDefinedServiceInterfaceDeployment

[TPS_MANI_03046] 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 is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	UserDefined configuration settings for an Event. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceEvent Deployment			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 7.13: UserDefinedEventDeployment

[TPS_MANI_03047] 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 is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedMethodDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	UserDefined configuration settings for a Method. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceMethod Deployment			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 7.14: UserDefinedMethodDeployment

[TPS_MANI_03048] 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 is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedFieldDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	UserDefined configuration settings for a Field. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceFieldDeployment			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table 7.15: UserDefinedFieldDeployment

7.2 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.

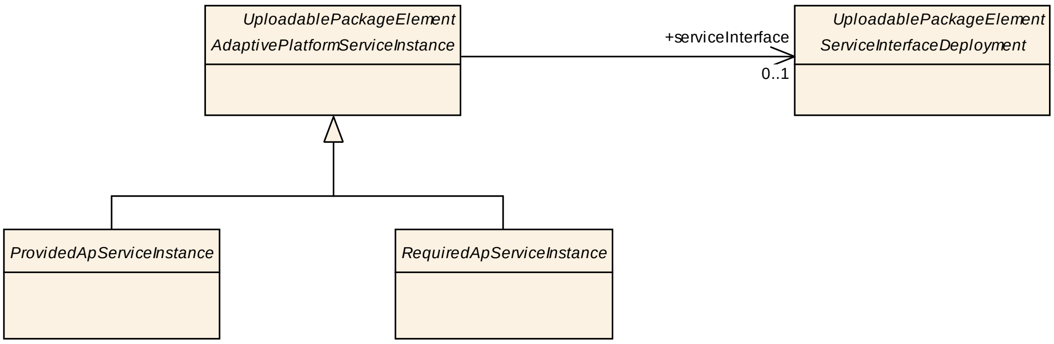

Figure 7.4: Modeling of the `AdaptivePlatformServiceInstance`

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::Deployment::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a service instance in an abstract way. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
e2eEventProtectionProps	End2EndEventProtectionProps	*	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 serviceInterface. **Tags:** atp.Status=draft
secureComConfig	ServiceInterfaceElementSecureComConfig	*	aggr	Configuration settings to secure the communication of ServiceInterface elements. **Tags:** atp.Status=draft
serviceInterface	ServiceInterfaceDeployment	0..1	ref	Reference to a ServiceInterfaceDeployment that identifies the ServiceInterface that is represented by the ServiceInstance. **Tags:** atp.Status=draft

Table 7.16: AdaptivePlatformServiceInstance

Class	**RequiredApServiceInstance (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in an abstract way. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 7.17: RequiredApServiceInstance

Class	**ProvidedApServiceInstance (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in an abstract way. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 7.18: ProvidedApServiceInstance

There are two alternative ways to relate a `AdaptivePlatformServiceInstance` with a `Machine` as described in [TPS_MANI_03000] and [TPS_MANI_03001]. Figure Figure 7.5 shows both approaches in an example.

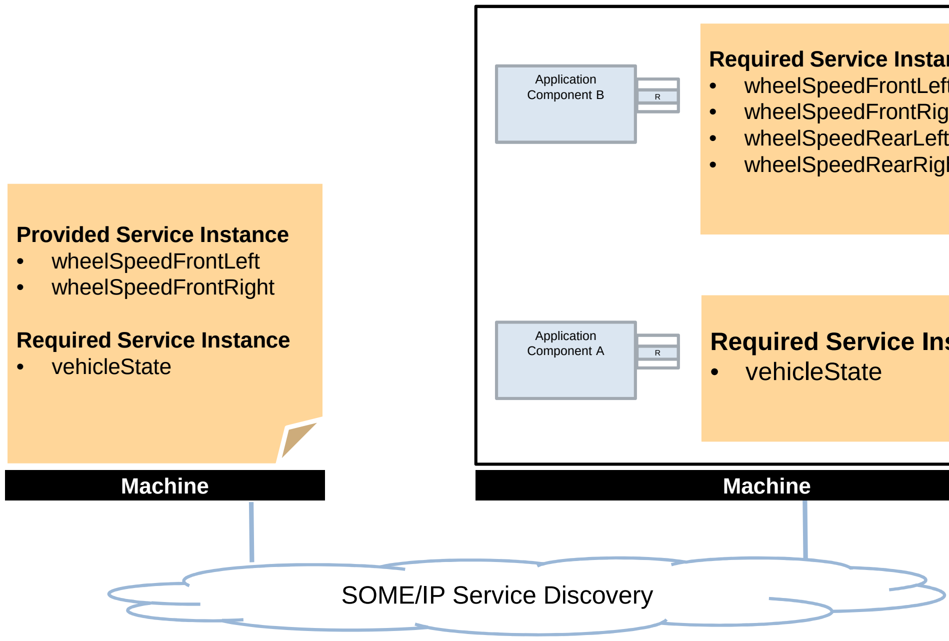

Figure 7.5: Different approaches for ServiceInstanceMapping

[TPS_MANI_03001] Mapping of AdaptivePlatformServiceInstance to a Machine ⌈ ServiceInstanceToMachineMapping is used to assign an AdaptivePlatformServiceInstance to (via a CommunicationConnector) a Machine. 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)*

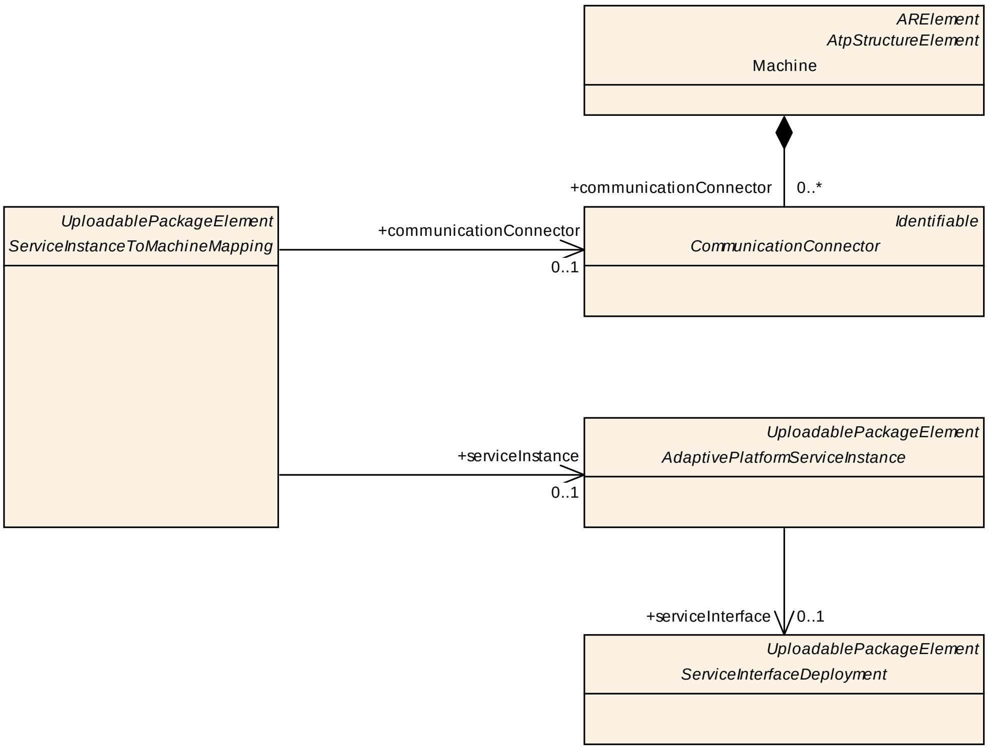

Figure 7.6: ServiceInstanceToMachineMapping

Class	**ServiceInstanceToMachineMapping (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstanceMapping			
Note	This meta-class represents the ability to map a AdaptivePlatformServiceInstance to a CommunicationConnector of a Machine. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
communicationConnector	CommunicationConnector	0..1	ref	Reference to the Machine to which the ServiceInstance is mapped. **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatformServiceInstance	0..1	ref	Reference to a ServiceInstance that is mapped to the Machine. **Tags:** atp.Status=draft

Table 7.19: ServiceInstanceToMachineMapping

[constr_3297] 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. ⌋()

[TPS_MANI_03000] 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, resp. the concrete subclasses RequiredApServiceInstance and ProvidedApServiceInstance.

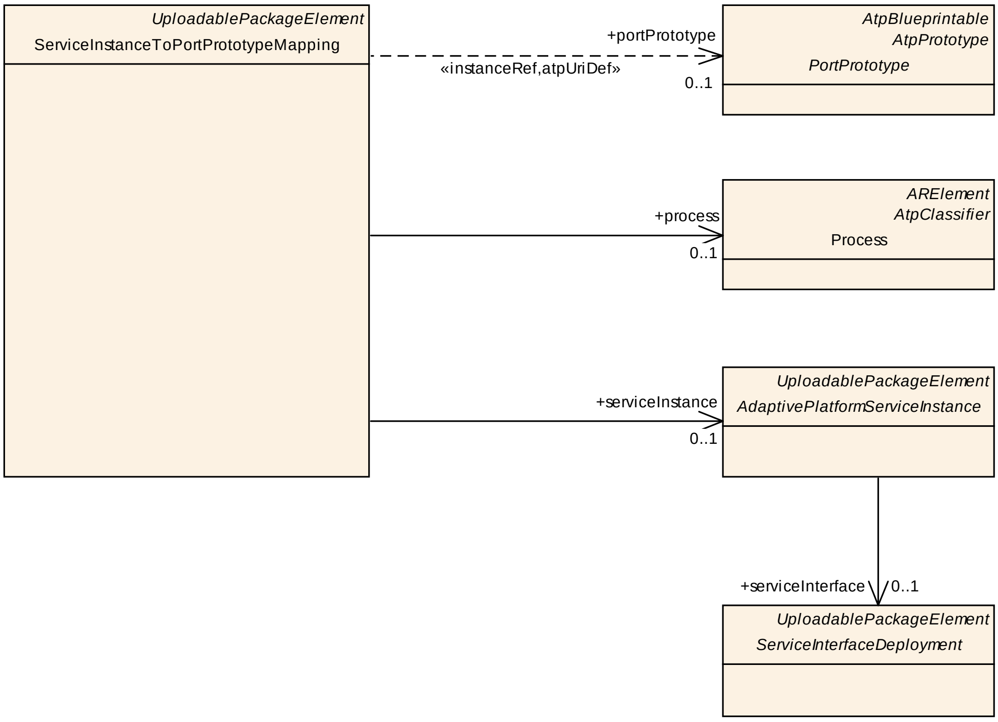

Figure 7.7: ServiceInstanceToPortPrototypeMapping

Class	**ServiceInstanceToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstanceMapping			
Note	This meta-class represents the ability to assign a transport layer dependent ServiceInstance to a PortPrototype. With this mapping it is possible to define how specific PortPrototypes are represented in the middleware in terms of service configuration. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInstanceToApplicationEndpointMappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	*Type*	*Mul.*	*Kind*	*Note*

portPrototype	PortPrototype	0..1	iref	Reference to a specific PortPrototypes that represents the ServiceInstance. **Tags:** atp.Status=draft
process	Process	0..1	ref	Reference to the Process in which the Executable that contains the SoftwareComponent and the referenced PortPrototype is executed. **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatformServiceInstance	0..1	ref	Reference to a ServiceInstance that is represented in the Software Component by the mapped group of PortPrototypes. **Tags:** atp.Status=draft

Table 7.20: 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.

7.2.1 SOME/IP Service Instance Deployment

In the case of SOME/IP used as the middleware the derived meta-classes are `ProvidedSomeipServiceInstance` resp. `RequiredSomeipServiceInstance`. These meta-classes also carry attributes that apply for the service discovery on SOME/IP.

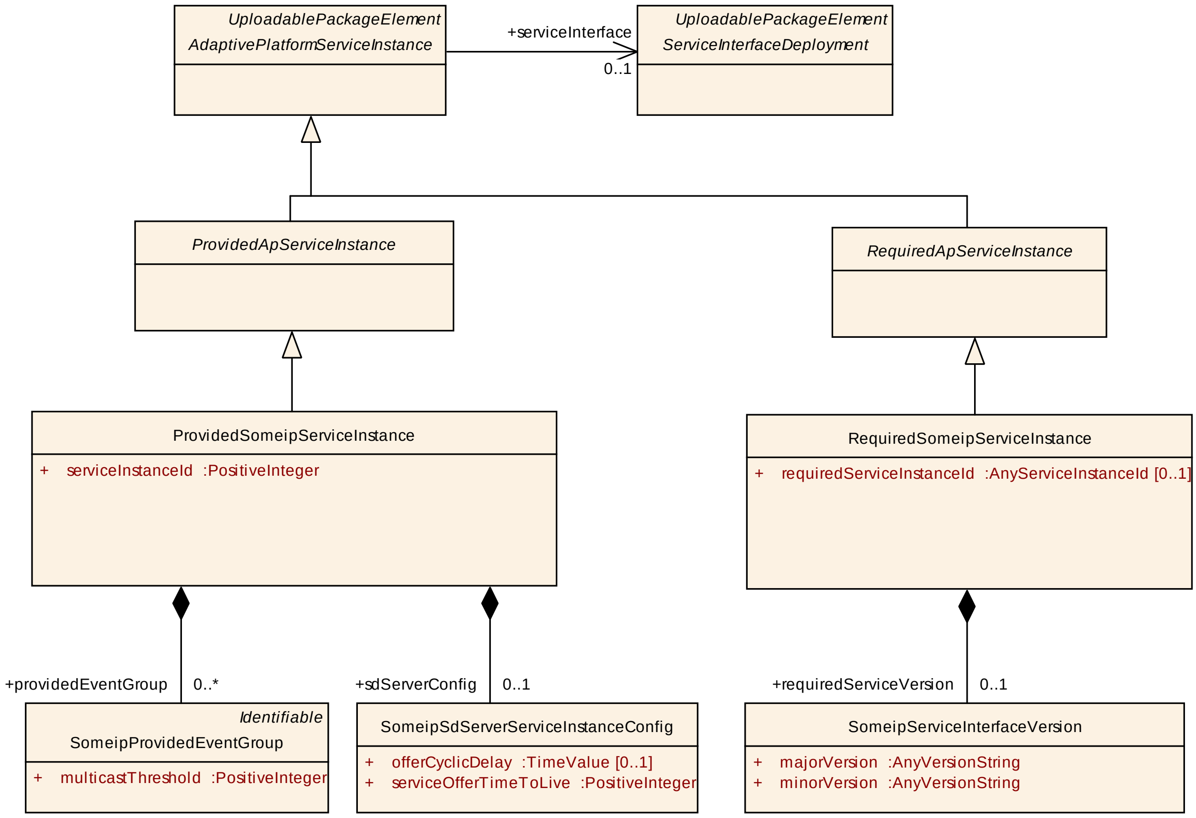

Figure 7.8: SOME/IP Service Instances

7.2.1.1 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::Deployment::ServiceInstance			
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.Status=draft; atp.recommendedPackage=ServiceInstances			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, ProvidedApServiceInstance, Referrable, UploadablePackageElement			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

providedEventGroup	SomeipProvidedEventGroup	*	aggr	List of EventGroups that are provided by the Service Instance. **Tags:** atp.Status=draft
sdServerConfig	SomeipSdServerServiceInstanceConfig	0..1	aggr	Server specific configuration settings relevant for the SOME/IP service discovery. **Tags:** atp.Status=draft
serviceInstanceId	PositiveInteger	1	attr	Identification number that is used by SOME/IP service discovery to identify the instance of the service.

Table 7.21: ProvidedSomeipServiceInstance

[constr_3287] 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.

7.2.1.1.1 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] IP configuration for a ProvidedSomeipServiceInstance ⌈ A ProvidedSomeipServiceInstance can be mapped to a CommunicationConnector of a Machine 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] ProvidedSomeipServiceInstance Fanout ⌈ It is allowed to map the same ProvidedSomeipServiceInstance to different CommunicationConnectors of a Machine. 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)*

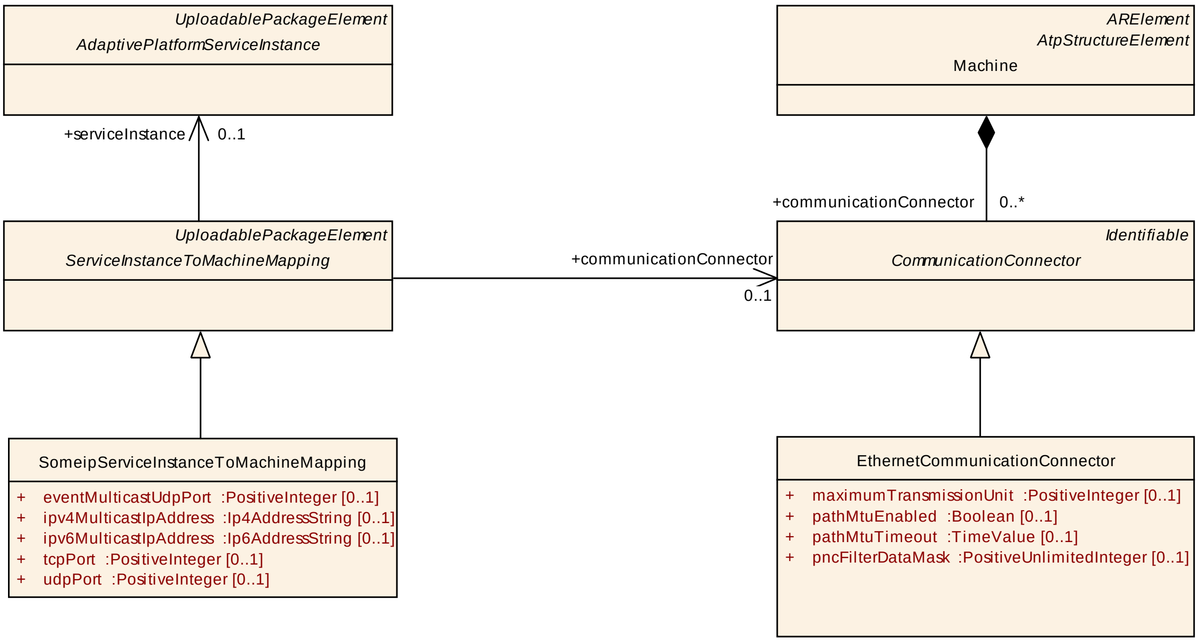

Figure 7.9: SomeipServiceInstanceToMachineMapping with TP and IP configuration

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, PackageableElement, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
baudrate	PositiveUnlimitedInteger	0..1	attr	Channels speed in bits/s.
physicalChannel	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 7.22: 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.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 7.23: CommunicationConnector

Class	**EthernetCommunicationConnector**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::Ethernet Topology			
Note	Ethernet specific attributes to the CommunicationConnector.			
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. This attribute should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs. Note that for one EcuInstance all contributing pncFilterDataMask will be bitwise ORed to obtain the value of UdpNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload (8 Byte) of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset.
unicastNetworkEndpoint	NetworkEndpoint	0..1	ref	Network Endpoint that defines the IPAddress of the machine. **Tags:** atp.Status=draft

Table 7.24: EthernetCommunicationConnector

[constr_3288] 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] IPv4 Multicast event destination address ⌈ Meta-class `SomeipServiceInstanceToMachineMapping` defines the multicast IPv4 address to which the `event`s and notification events are send with the attribute `ipv4MulticastIpAddress`. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03061] IPv6 Multicast event destination address ⌈ Meta-class `SomeipServiceInstanceToMachineMapping` defines the multicast IPv6 address to which the `event`s and notification events are sent with the attribute `ipv6MulticastIpAddress`. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03005] 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] 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	**NetworkEndpoint**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::Ethernet Topology			
Note	The network endpoint defines the network addressing (e.g. IP-Address or MAC multicast address).			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
fullyQualifiedDomainName	String	0..1	attr	Defines the fully qualified domain name (FQDN) e.g. some.example.host.
infrastructureServices	InfrastructureServices	0..1	aggr	Defines the network infrastructure services provided or consumed.
networkEndpointAddress	NetworkEndpointAddress	1..*	aggr	Definition of a Network Address. **Tags:** xml.namePlural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Priority of this Network-Endpoint.

Table 7.25: NetworkEndpoint

Class	**NetworkEndpointAddress (abstract)**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::Ethernet Topology			
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.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 7.26: NetworkEndpointAddress

Class	**Ipv4Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::Ethernet Topology			
Note	Internet Protocol version 4 (IPv4) configuration.			
Base	ARObject, NetworkEndpointAddress			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

assignmentPriority	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.
defaultGateway	Ip4AddressString	0..1	attr	IP address of the default gateway.
dnsServerAddress	Ip4AddressString	*	attr	IP addresses of preconfigured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
ipAddressKeepBehavior	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 ipv4AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv4AddressSource	Ipv4AddressSourceEnum	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 7.27: Ipv4Configuration

Class	**Ipv6Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 6 (IPv6) configuration.			
Base	ARObject, NetworkEndpointAddress			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
assignmentPriority	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.
dnsServerAddress	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)

ipAddressKeepBehavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipAddressPrefixLength	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 ipv6AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv6AddressSource	Ipv6AddressSourceEnum	0..1	attr	Defines how the node obtains its IP address.

Table 7.28: Ipv6Configuration

7.2.1.1.2 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] 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] 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] 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] 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] 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.

⌋*()*

7.2.1.1.3 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.

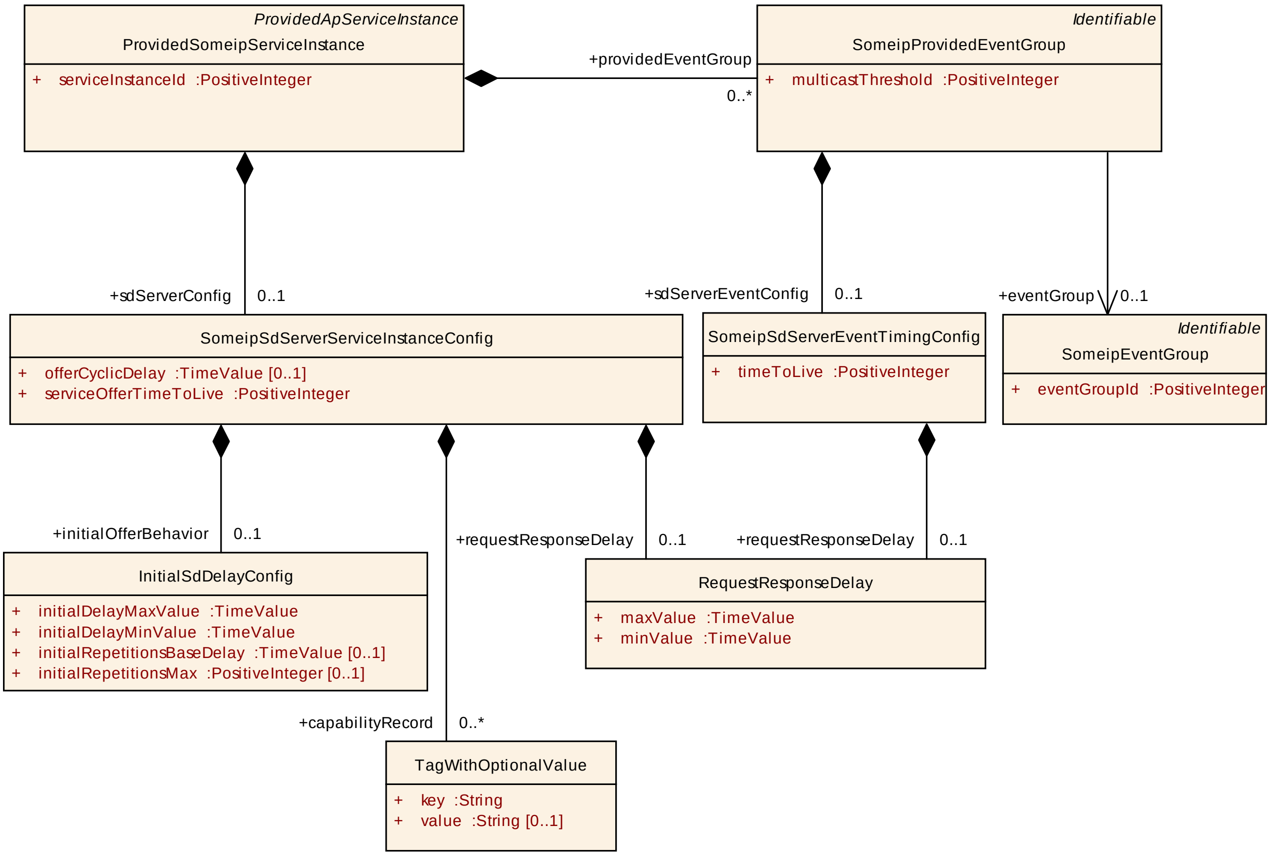

Figure 7.10: SOME/IP Service Discovery Server configuration settings

For every ProvidedSomeipServiceInstance on a Server different phases are existing:

- Down
- Available
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase

[TPS_MANI_03011] Server Timing configuration for a ProvidedSomeipServiceInstance ⌈ The Server Timing is configurable with SomeipSdServerServiceInstanceConfig that is aggregated in the role sdServerConfig by the ProvidedSomeipServiceInstance for which the Timing is valid. ⌋ *(RS_MANI_00024)*

Class	**SomeipSdServerServiceInstanceConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
Note	Server specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
capabilityRecord	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Tags:** atp.Status=draft
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 7.29: SomeipSdServerServiceInstanceConfig

[TPS_MANI_03012] 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] 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] 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.			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
initialDelayMaxValue	TimeValue	1	attr	Max Value in seconds to delay randomly the first offer (if aggregated by SdServerConfig) or the transmission of a find message (if aggregated by SdClientConfig).
initialDelayMinValue	TimeValue	1	attr	Min Value in seconds to delay randomly the first offer (if aggregated by SdServerConfig) or the transmission of a find message (if aggregated by SdClientConfig).
initialRepetitionsBaseDelay	TimeValue	0..1	attr	The base delay for offer repetitions (if aggregated by SdServerConfig) or find repetitions (if aggregated by SdClientConfig. Successive find messages have an exponential back off delay.
initialRepetitionsMax	PositiveInteger	0..1	attr	Describes the maximum amount of offer repetitions (if aggregated by SdServerConfig) or the maximum amount of find repetitions (if aggregated by SdClientConfig).

Table 7.30: InitialSdDelayConfig

[TPS_MANI_03015] 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] 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::Ethernet Topology			
Note	Time to wait before answering the query.			
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 7.31: RequestResponseDelay

Figure 7.11 shows an example of the different SOME/IP phases on the Server side.

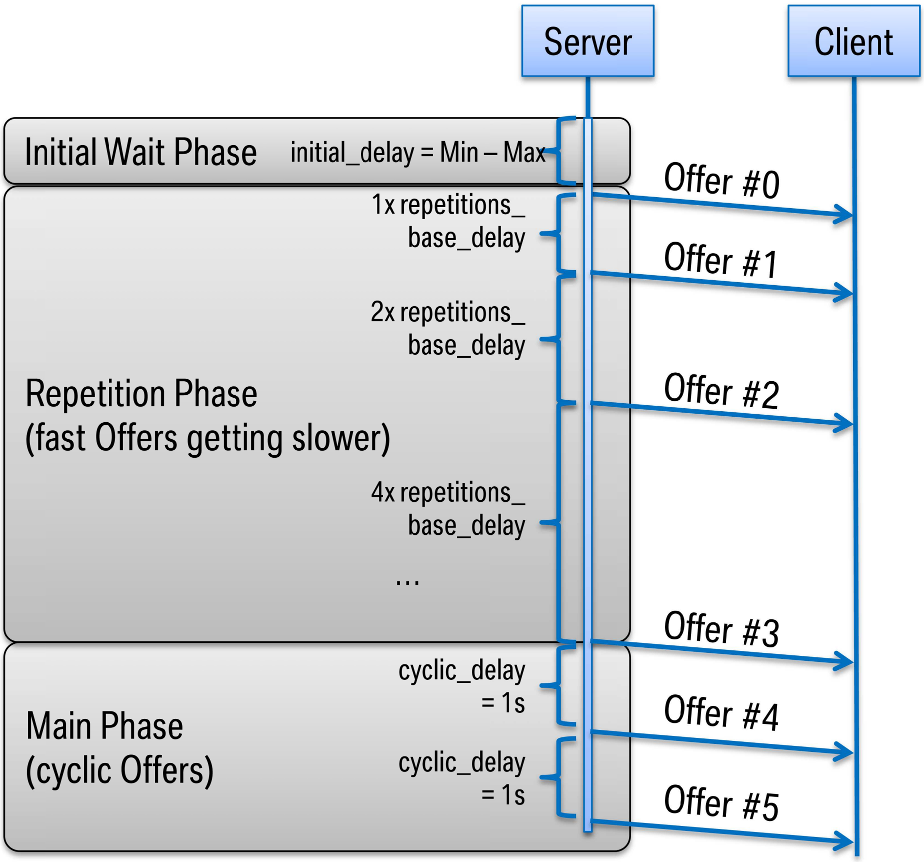

Figure 7.11: SOME/IP Server Timing example

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] 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.			
Base	ARObject			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
key	String	1	attr	Defines a key.
value	String	0..1	attr	Defines the corresponding value.

Table 7.32: TagWithOptionalValue

7.2.1.1.4 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::Deployment::ServiceInstance			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the provided side for each EventGroup separately. **Tags:** 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 EventGroup settings are valid. **Tags:** atp.Status=draft
multicastThreshold	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.

sdServerEventConfig	SomeipSdServerEventTimingConfig	0..1	aggr	Server Timing configuration settings that are EventGroup specific. **Tags:** atp.Status=draft

Table 7.33: SomeipProvidedEventGroup

[TPS_MANI_03018] 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)*

[TPS_MANI_03019] TTL for SubscribeEventGroupAck Entries ⌈ The lifetime of a event subscription is configurable with the timeToLive attribute of SomeipSdServerEventTimingConfig.

If the time that is configured by timeToLive expires the event subscription is canceled. ⌋*(RS_MANI_00024)*

[TPS_MANI_03020] 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	**SomeipSdServerEventTimingConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
Note	EventGroup specific timing configuration settings. **Tags:** atp.Status=draft			
Base	ARObject			
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 eventGroup is valid. This value is sent from the server to the client in the SD subscribeEventGroupAck message.

Table 7.34: SomeipSdServerEventTimingConfig

7.2.1.2 Required Service Instance

[TPS_MANI_03059] 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::Deployment::ServiceInstance			
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.Status=draft; atp.recommendedPackage=ServiceInstances			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, RequiredApServiceInstance, UploadablePackageElement			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
requiredEventGroup	SomeipRequiredEventGroup	*	aggr	List of EventGroups that are used by the RequiredServiceInstance. **Tags:** atp.Status=draft
requiredServiceInstanceId	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID.
requiredServiceVersion	SomeipServiceInterfaceVersion	0..1	aggr	This element is used to configure for which version (major version/minor version) of the SomeIp Service the Service Discovery will search. **Tags:** atp.Status=draft
sdClientConfig	SomeipSdClientServiceInstanceConfig	0..1	aggr	Client specific configuration settings relevant for the SOME/IP service discovery. **Tags:** atp.Status=draft

Table 7.35: RequiredSomeipServiceInstance

[constr_3293] Mandatory information of a RequiredSomeipServiceInstance ⌈ The RequiredSomeipServiceInstance shall always define the attributes requiredServiceInstanceId and requiredServiceVersion. ⌋ *()*

[TPS_MANI_03021] Requirements on the service version from the client's point of view ⌈ The meta-class RequiredSomeipServiceInstance can also make further specifications regarding the version of the service from the client's point of view.

For this purpose, the attribute RequiredSomeipServiceInstance.requiredServiceVersion exists and provides the ability to define the required major version (SomeipServiceInterfaceVersion.majorVersion) and the minor version (SomeipServiceInterfaceVersion.minorVersion). ⌋*(RS_MANI_00024)*

Class	**SomeipServiceInterfaceVersion**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
Note	This meta-class represents the ability to describe a version of a SOME/IP ServiceInterface. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
majorVersion	AnyVersionString	1	attr	Major Version of the ServiceInterface. Value can be set to a number that represents the Major Version of the searched service or to ANY.
minorVersion	AnyVersionString	1	attr	Minor Version of the ServiceInterface. Value can be set to a number that represents the Minor Version of the searched service or to ANY.

Table 7.36: SomeipServiceInterfaceVersion

7.2.1.2.1 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] Context of RequiredSomeipServiceInstance ⌈ A RequiredSomeipServiceInstance can be mapped to a CommunicationConnector of a Machine 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)*

7.2.1.2.2 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.

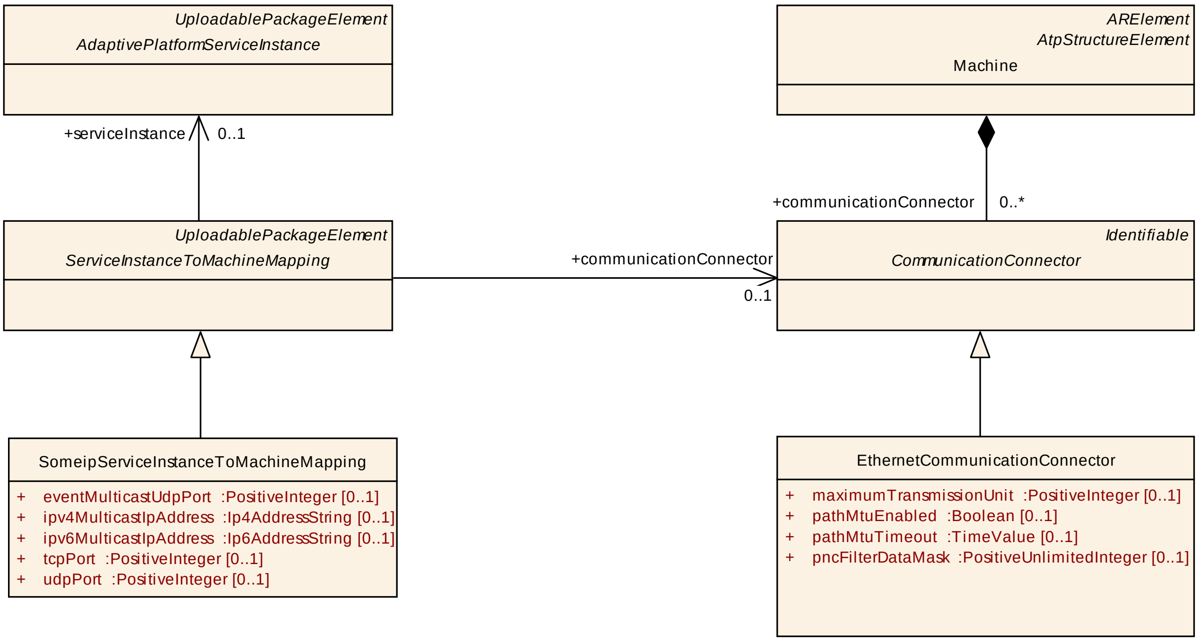

Figure 7.12: SomeipServiceInstanceToMachineMapping with TP and IP configuration

[TPS_MANI_03023] 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] 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] 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] 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].

7.2.1.2.3 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] Client Timing configuration for a RequiredSomeipServiceInstance ⌈ The Client Timing is configurable with SomeipSdClientServiceInstanceConfig that is aggregated in the role sdClientConfig by the RequiredSomeipServiceInstance for which the Timing is valid. ⌋*(RS_MANI_00024)*

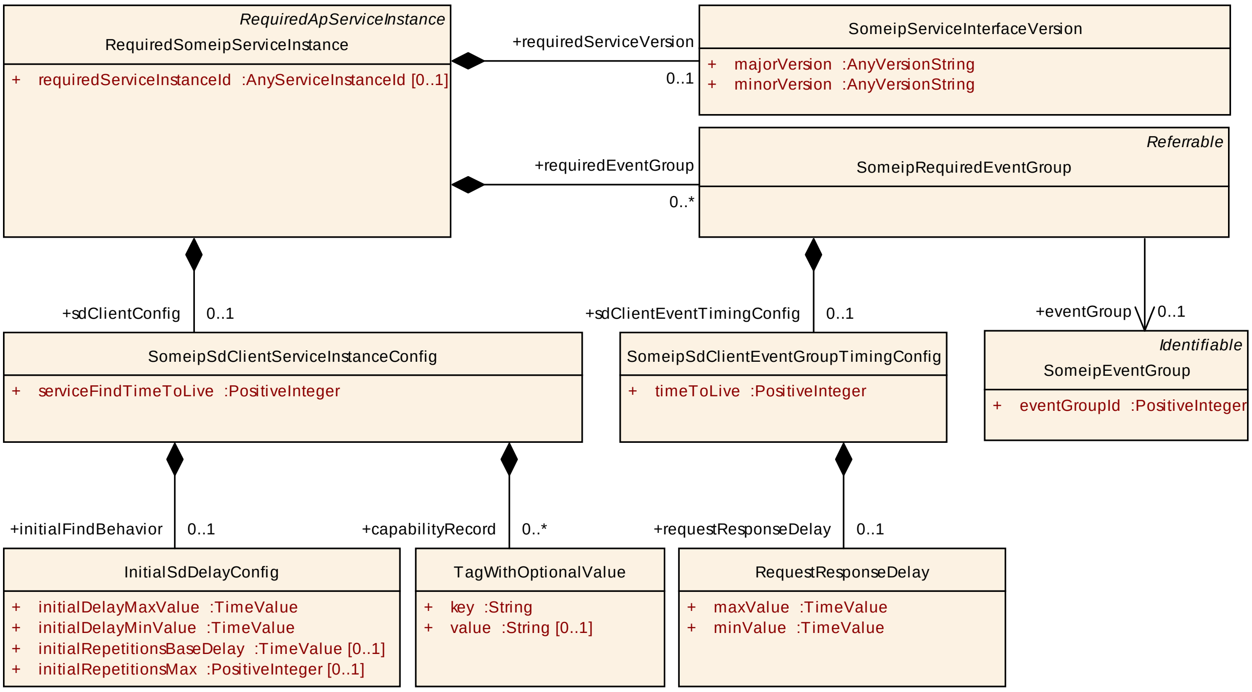

Figure 7.13: SOME/IP Service Discovery Client configuration settings

Class	**SomeipSdClientServiceInstanceConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
Note	Client specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
capabilityRecord	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Tags:** atp.Status=draft
initialFindBehavior	InitialSdDelayConfig	0..1	aggr	Controls initial find behavior of clients. **Tags:** atp.Status=draft
serviceFindTimeToLive	PositiveInteger	1	attr	This attribute represents the ability to define the time in seconds the service find is valid.

Table 7.37: SomeipSdClientServiceInstanceConfig

[TPS_MANI_03026] 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] 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] 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 7.14 shows an example of the different SOME/IP phases on the Client side.

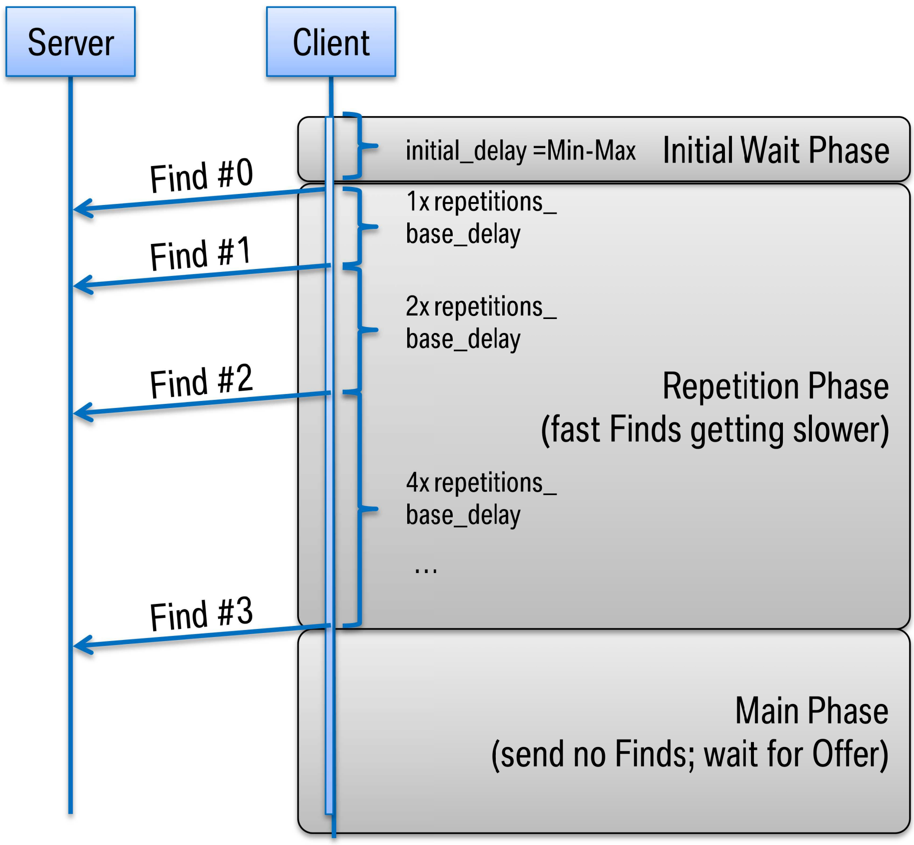

Figure 7.14: SOME/IP Client Timing example

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] 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)*

7.2.1.2.4 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::Deployment::ServiceInstance			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the required side for each EventGroup separately. **Tags:** 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 EventGroup settings are valid. **Tags:** atp.Status=draft
sdClientEventTimingConfig	SomeipSdClientEventGroupTimingConfig	0..1	aggr	Client Timing configuration settings that are EventGroup specific. **Tags:** atp.Status=draft

Table 7.38: SomeipRequiredEventGroup

Class	**SomeipSdClientEventGroupTimingConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
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.Status=draft			
Base	ARObject			
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 7.39: SomeipSdClientEventGroupTimingConfig

[TPS_MANI_03030] **`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] 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)

7.2.2 User Defined Service Instance Deployment

[TPS_MANI_03032] 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)

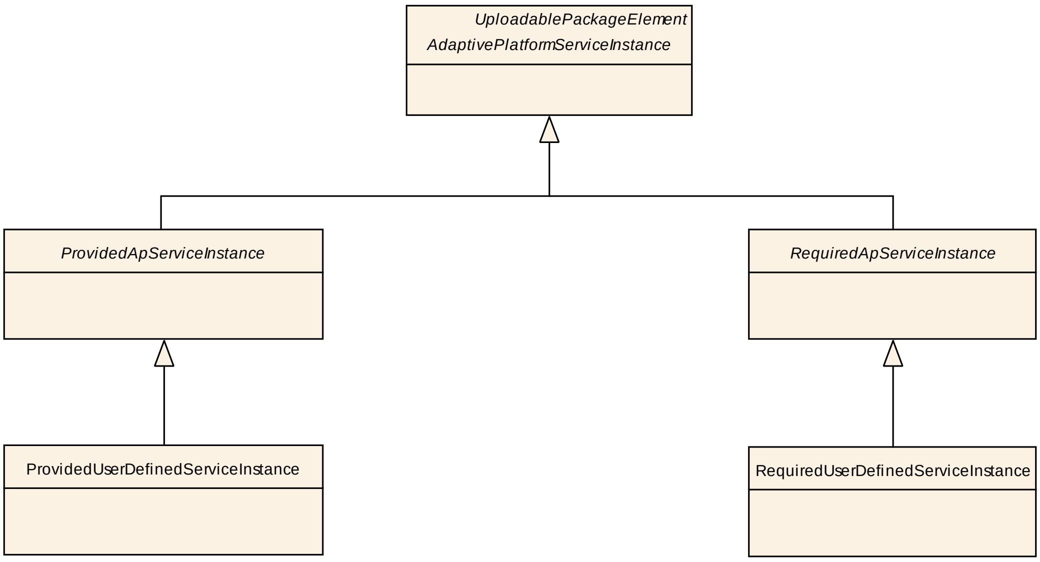

Figure 7.15: User Defined Service Instance Deployment

Class	**ProvidedUserDefinedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
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.Status=draft; atp.recommendedPackage=ServiceInstances			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, ProvidedApServiceInstance, Referrable, UploadablePackageElement			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 7.40: ProvidedUserDefinedServiceInstance

Class	**RequiredUserDefinedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance			
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.Status=draft; atp.recommendedPackage=ServiceInstances			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, RequiredApServiceInstance, UploadablePackageElement			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 7.41: 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.

7.3 EndToEndProtection

AUTOSAR supports the protection of events and Field notifiers with E2E Profiles that are defined in the E2E Communication Protection Library [16].

[TPS_MANI_03127] Usage of End2EndEventProtectionProps ⌈ The End2EndEventProtectionProps element is used to define event or notifier specific E2E configuration settings in the context of an AdaptivePlatformServiceInstance. ⌋*(RS_MANI_00028)*

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] E2E profile ⌈ The E2E profile is defined by E2EProfileConfiguration.profileName. ⌋*(RS_MANI_00028)*

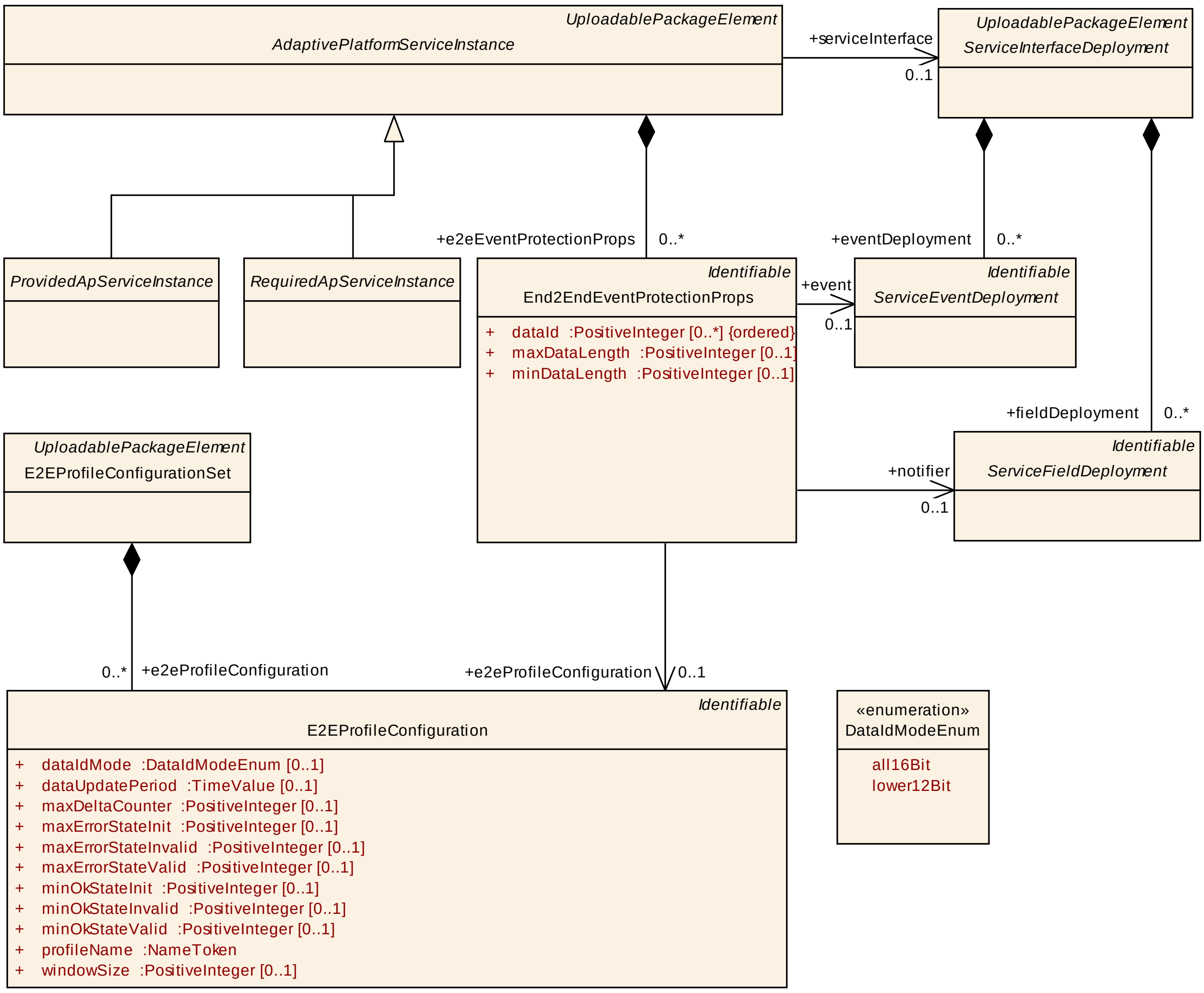

Figure 7.16: E2E EventProtection

[TPS_MANI_03130] 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] 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)*

[constr_3380] `End2EndEventProtectionProps` shall not reference an `event` and a `notifier` at the same time ⌈ The `End2EndEventProtectionProps` element shall reference either an `event` or a `notifier`. ⌋*()*

[TPS_MANI_03128] 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 or notifier. ⌋*(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::Deployment::E2E			
Note	This element allows to protect an event or a field notifier with an E2E profile. **Tags:** 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.
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 Data in bits.
minDataLength	PositiveInteger	0..1	attr	Minimum length of Data in bits.
notifier	ServiceFieldDeployment	0..1	ref	Reference to a field notifier that is protected by an E2E profile. **Tags:** atp.Status=draft

Table 7.42: End2EndEventProtectionProps

Class	E2EProfileConfigurationSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::E2E			
Note	This meta-class represents the ability to aggregate a collection of E2EProfileConfigurations. **Tags:** 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 7.43: E2EProfileConfigurationSet

Class	E2EProfileConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::E2E			
Note	This element holds E2E profile specific configuration settings. **Tags:** 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.
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.
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 7.44: 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 7.45: 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 [10].

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.

7.4 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] 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. And it allows to protect different elements of a ServiceInterface with different security protection mechanisms.

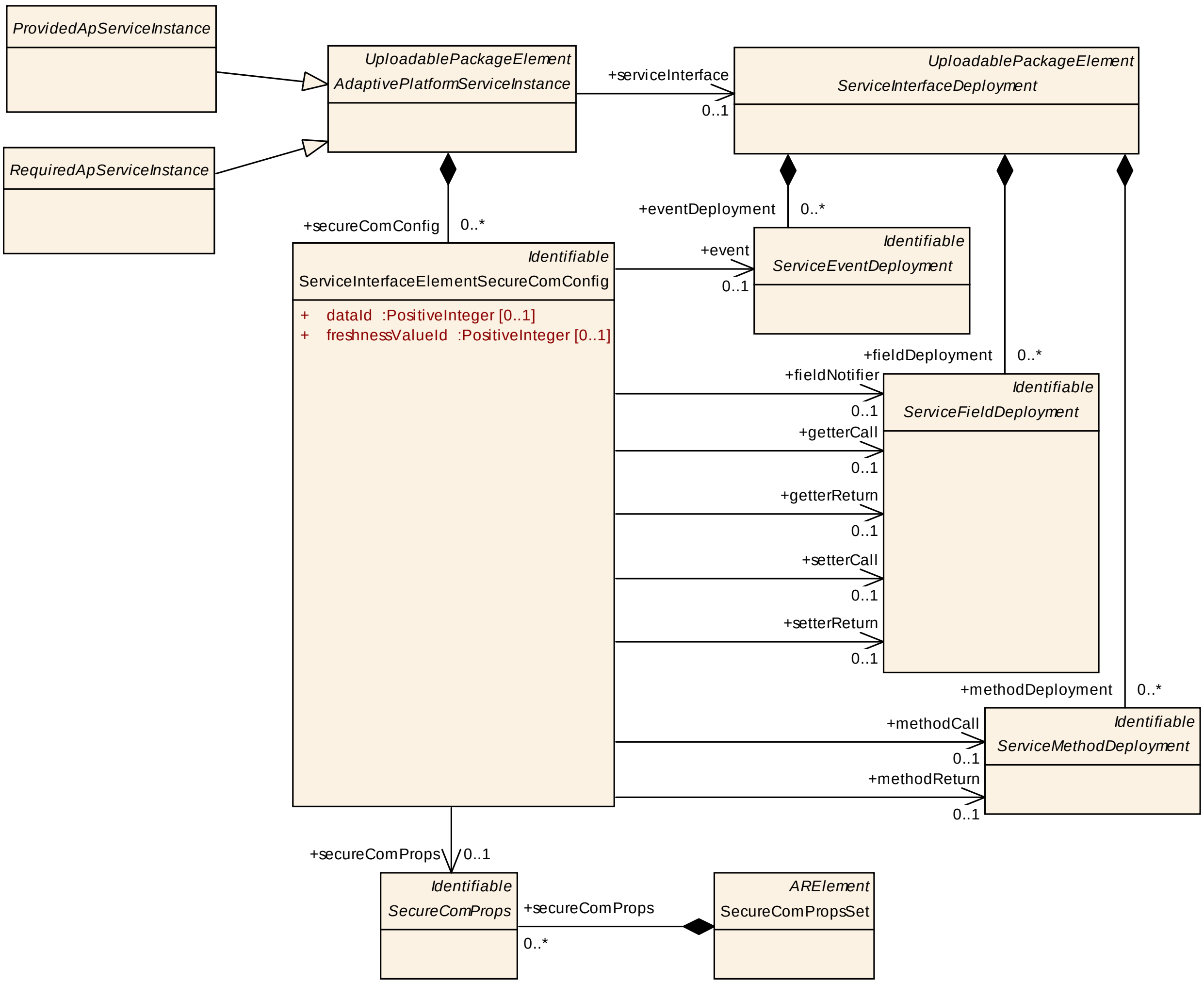

Figure 7.17: Secure Communication

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::Deployment::SecureCommunication			
Note	This element allows to secure the communication of the referenced ServiceInterface element. **Tags:** 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
fieldNotifier	ServiceFieldDeployment	0..1	ref	Reference to a field notifier that is protected by a security protocol. **Tags:** atp.Status=draft
freshnessValueId	PositiveInteger	0..1	attr	This attribute defines the Id of the Freshness Value.
getterCall	ServiceFieldDeployment	0..1	ref	Reference to a field getter call message that is protected by a security protocol. **Tags:** atp.Status=draft
getterReturn	ServiceFieldDeployment	0..1	ref	Reference to a field getter return message that is protected by a security protocol. **Tags:** atp.Status=draft
methodCall	ServiceMethodDeployment	0..1	ref	Reference to a method call message that is protected by a security protocol. **Tags:** atp.Status=draft
methodReturn	ServiceMethodDeployment	0..1	ref	Reference to a method return message that is protected by a security protocol. **Tags:** atp.Status=draft
secureComProps	SecureComProps	0..1	ref	Reference to the communication security protocol and its configuration settings that will provide communication security for the referenced ServiceInterfaceElement that is exchanged between a ProvidedServiceInstance and one or several RequiredServiceInstances. **Tags:** atp.Status=draft

setterCall	ServiceFieldDeployment	0..1	ref	Reference to a field setter call message that is protected by a security protocol. **Tags:** atp.Status=draft
setterReturn	ServiceFieldDeployment	0..1	ref	Reference to a field setter return message that is protected by a security protocol. **Tags:** atp.Status=draft

Table 7.46: ServiceInterfaceElementSecureComConfig

[constr_3391] `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 configuration settings that are specific for the referenced `ServiceInterface` element. In addition the `ServiceInterfaceElementSecureComConfig` references the `SecureComProps` meta-class and defines with this reference the security protocol that will be used for the protection. The security protocol configuration settings that are defined in `SecureComProps` are not `ServiceInterface` element specific and may be reused by several `ServiceInterfaceElementSecureComConfig` elements.

Class	**SecureComPropsSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	This meta-class represents the ability to aggregate a collection of SecureComProps.. **Tags:** atp.Status=draft; atp.recommendedPackage=SecureComPropsSets			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
secureComProps	SecureComProps	*	aggr	This represents the collection of SecureComProps aggregated at the SecureComPropsSet. **Tags:** atp.Status=draft

Table 7.47: SecureComPropsSet

Class	**SecureComProps (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	This meta-class defines a communication security protocol and its configuration settings. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 7.48: SecureComProps

7.4.1 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`.

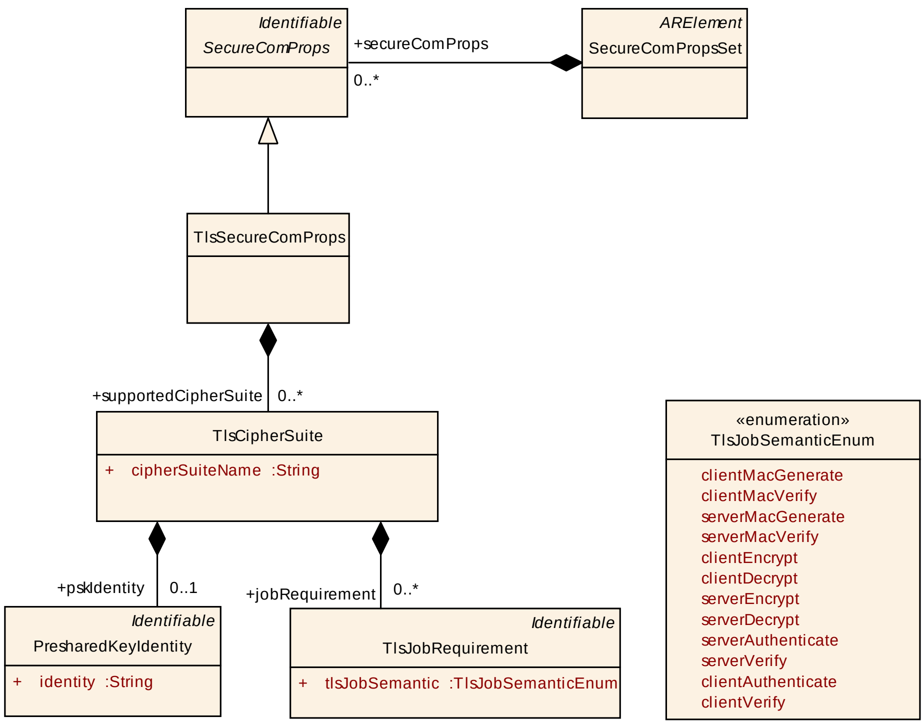

Figure 7.18: Secure Communication over TLS

Class	**TlsSecureComProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	Configuration of the Transport Layer Security protocol (TLS). **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, SecureComProps			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

supportedCipherSuite	TlsCipherSuite	*	aggr	Collection of supported cipher suites that are used to negotiate the security settings for a network connection. **Tags:** atp.Status=draft

Table 7.49: TlsSecureComProps

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 AdaptivePlatformServiceInstances 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.

[TPS_MANI_03134] Configuration of supported TLS ciphersuites ⌈ The supported TLS ciphersuites are configured on an AdaptivePlatformServiceInstance via ServiceInterfaceElementSecureComConfig with TlsCipherSuite elements that are aggregated by TlsSecureComProps in the role supportedCipherSuite. Each TlsCipherSuite element contains the cipherSuiteName attribute that describes the ciphersuite. ⌋*(RS_MANI_00036)*

Class	**TlsCipherSuite**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	This meta-class defines a cipher suite that is supported in TLS. It defines a named combination of authentication and encryption algorithms used to negotiate the security settings for a network connection that uses the Transport Layer Security. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
cipherSuiteName	String	1	attr	This attributes defines the CipherSuite name. e.g. "TLS_RSA_WITH_RC4_128_MD5".
jobRequirement	TlsJobRequirement	*	aggr	Collection of cryptographic job requirements. **Tags:** atp.Status=draft
pskIdentity	PresharedKeyIdentity	0..1	aggr	Configuration of TLS-PSK identity that will be send from the client to the server. **Tags:** atp.Status=draft

Table 7.50: TlsCipherSuite

If the client and the server are able to negotiate a cipher, and the client accepts the certificate provided by the server then the client will initiate the key exchange. The client indicates which key to use by sending a PSK identity to the server.

[TPS_MANI_03135] Configuration of TLS PSK Identity ⌈ The TLS PSK Identity is configured with the `PresharedKeyIdentity.identity` attribute. ⌋ *(RS_MANI_00036)*

Class	**PresharedKeyIdentity**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	TLS-PSK are symmetric keys that are shared in advance among the communicating parties, to establish a TLS connection. The client indicates which key to use by sending a PSK identity to the server. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
identity	String	1	attr	This attribute defines the PSK identity that is sent from the client to the server to indicate which PreSharedKey will be used for the handshake.

Table 7.51: PresharedKeyIdentity

Please note that it is of course allowed that the different `ServiceInterfaceElementSecureComConfig` elements of a `RequiredApServiceInstance` point to the same `TlsSecureComProps` to indicate that the same secure channel is used for the complete outgoing service communication.

After the successful handshake the encryption and/or authentication of the data that is referenced by the `ServiceInterfaceElementSecureComConfig` will be provided. The cryptographic jobs that need to be supported by the communication partners are defined by `TlsJobRequirement`.

[TPS_MANI_03136] Configuration of requirements for the TLS cryptographic job ⌈ The TLS Job requirements are configured with the `TlsJobRequirement.tlsJobSemantic` attribute. ⌋ *(RS_MANI_00036)*

Class	**TlsJobRequirement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	Requirements for the cryptographic job that need to be executed. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
tlsJobSemantic	TlsJobSemanticEnum	1	attr	This attribute defines the cryptographic algorithm that needs to be supported.

Table 7.52: TlsJobRequirement

Enumeration	**TlsJobSemanticEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication
Note	List of cryptographic routines supported by TLS. **Tags:** atp.Status=draft
Literal	***Description***
clientAuthen-ticate	Client supports the generation of a Signature. **Tags:** atp.EnumerationValue=10
clientDecrypt	Client supports decryption. **Tags:** atp.EnumerationValue=5
clientEncrypt	Client supports encryption. **Tags:** atp.EnumerationValue=4
clientMac Generate	Client supports the generation of a Message Authentication Code **Tags:** atp.EnumerationValue=0
clientMac Verify	Client supports the verification of a Message Authentication Code **Tags:** atp.EnumerationValue=1
clientVerify	Client supports the verification of a Signature. **Tags:** atp.EnumerationValue=11
serverAu-thenticate	Server supports the generation of a Signature. **Tags:** atp.EnumerationValue=8
serverDe-crypt	Server supports decryption. **Tags:** atp.EnumerationValue=7
serverEn-crypt	Server supports encryption. **Tags:** atp.EnumerationValue=6
serverMac Generate	Server supports the generation of a Message Authentication Code **Tags:** atp.EnumerationValue=2
serverMac Verify	Server supports the verification of a Message Authentication Code **Tags:** atp.EnumerationValue=3
serverVerify	Server supports the verification of a Signature. **Tags:** atp.EnumerationValue=9

Table 7.53: TlsJobSemanticEnum

[TPS_MANI_03137] `ServiceInterfaceElementSecureComConfig.dataId` and `ServiceInterfaceElementSecureComConfig.freshnessValueId` are not relevant in case of TLS communication ⌈ The attributes `ServiceInterfaceElementSecureComConfig.dataId` and `ServiceInterfaceElementSecureComConfig.freshnessValueId` are not relevant in case that the

ServiceInterfaceElementSecureComConfig refers to TlsSecureComProps. ⌋(*RS_MANI_00036*)

7.4.2 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 element needs to point to SecOcSecureComProps that contains the relevant configuration settings.

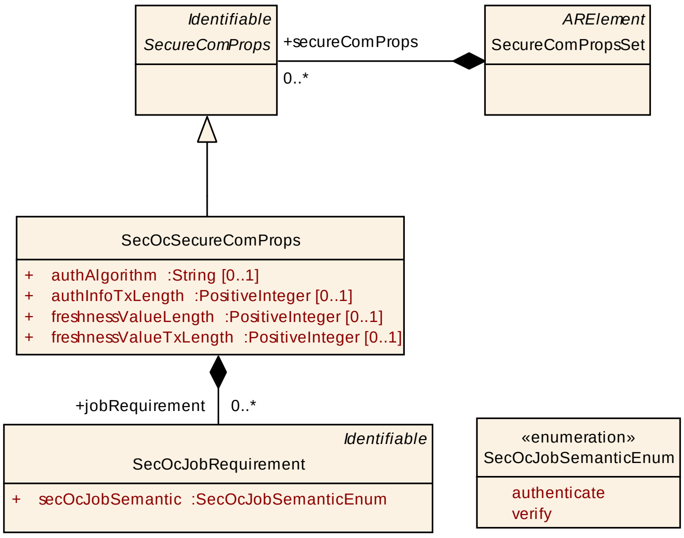

Figure 7.19: Secure Communication over SecOC

[constr_3392] ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication ⌈ The attributes ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case that ServiceInterfaceElementSecureComConfig refers to SecOcSecureComProps. ⌋ *()*

[TPS_MANI_03138] SecOC Security Profile ⌈ The SecOC security profile is defined by SecOcSecureComProps.category. ⌋(*RS_MANI_00036*)

Class	**SecOcSecureComProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	Configuration of AUTOSAR SecOC. **Tags:** 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 7.54: SecOcSecureComProps

[TPS_MANI_03139] 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 [10].

[TPS_MANI_03140] 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::Deployment::SecureCommunication			
Note	Requirements for the cryptographic job that need to be executed. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
secOcJobSemantic	SecOcJobSemanticEnum	1	attr	This attribute defines the cryptographic algorithm that needs to be supported.

Table 7.55: SecOcJobRequirement

Enumeration	**SecOcJobSemanticEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::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 7.56: SecOcJobSemanticEnum

8 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.

The Machine is able to aggregate one or several Processors. And each Processor consists of one or several ProcessorCores.

In addition the Machine meta-class allows to describe the available network connections and the configuration settings of the *Adaptive AUTOSAR Software Stack* that is running on this Machine.

An overview of the Machine meta-class is sketched in Figure 8.1.

[TPS_MANI_03035] Content of the Machine configuration ⌈ The purpose of the Machine is to provide machine specific configuration settings. ⌋*(RS_MANI_00018, RS_MANI_00020, RS_MANI_00021, RS_MANI_00022, RS_MANI_00023)*

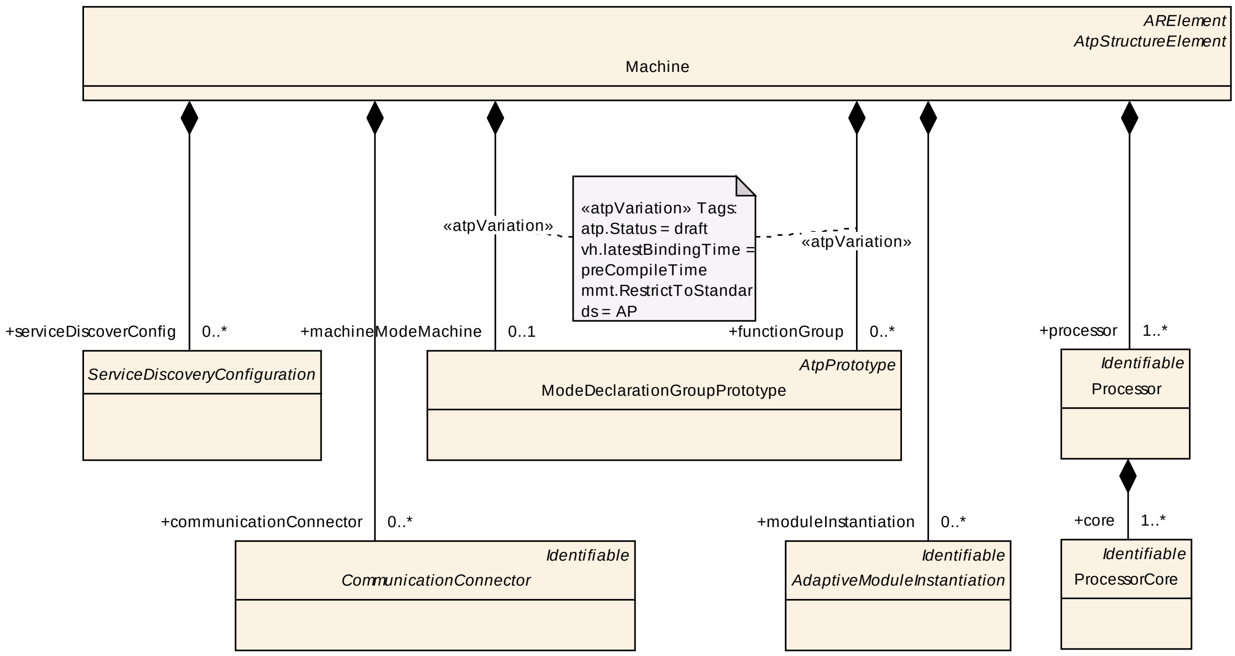

Figure 8.1: Overview about the content of the Machine configuration

Class	**Machine**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Machine			
Note	Machine that represents an Adaptive Autosar Software Stack. **Tags:** atp.Status=draft; atp.recommendedPackage=Machines			
Base	ARElement, ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Collectable Element, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

communicationConnector	CommunicationConnector	*	aggr	This aggregation defines the network connection of the machine. **Tags:** atp.Status=draft
defaultApplicationTimeout	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
functionGroup	ModeDeclarationGroupPrototype	*	aggr	This aggregation represents the collection of function groups of the enclosing Machine. **Stereotypes:** atpVariation **Tags:** 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
machineModeMachine	ModeDeclarationGroupPrototype	0..1	aggr	Set of MachineStates (Modes) that are defined for the machine. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=preCompileTime
moduleInstantiation	AdaptiveModuleInstantiation	*	aggr	Configuration of Adaptive Autosar module instances that are running on the machine. **Tags:** atp.Status=draft
perStateTimeout	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
secureCommunicationDeployment	SecureCommunicationDeployment	*	aggr	Deployment of secure communication protocol configuration settings to crypto module entities. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel; atp.Status=draft
serviceDiscoverConfig	ServiceDiscoveryConfiguration	*	aggr	Set of service discovery configuration settings that are defined on the machine for individual CommunicationConnectors. **Tags:** atp.Status=draft

Table 8.1: Machine

Class	**Processor**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Machine			
Note	This represents a processor for the execution of an AUTOSAR adaptive platform **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
core	ProcessorCore	1..*	aggr	This represents the collection of cores onwed by the enclosing processor. **Tags:** atp.Status=draft

Table 8.2: Processor

Class	**ProcessorCore**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Machine			
Note	This meta-class represents the ability to model a processor core for the execution of an AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 8.3: ProcessorCore

8.1 Network connection

One of the most prominent information defined in the context of the `Machine` 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 `Machine` 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 `Machine` 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	A physical channel is the transmission medium that is used to send and receive information between communicating ECUs. Each CommunicationCluster has at least one physical channel. Bus systems like CAN and LIN only have exactly one PhysicalChannel. A FlexRay cluster may have more than one PhysicalChannels that may be used in parallel for redundant communication. An ECU is part of a cluster if it contains at least one controller that is connected to at least one channel of the cluster.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
commConnector	CommunicationConnector	1..*	ref	Reference to the ECUInstance via a CommunicationConnector 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
frameTriggering	FrameTriggering	*	aggr	One frame triggering is defined for exactly one channel. Channels may have assigned an arbitrary number of frame triggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings must be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
iSignalTriggering	ISignalTriggering	*	aggr	One ISignalTriggering is defined for exactly one channel. Channels may have assigned an arbitrary number of ISignaltriggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings must be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild

pduTriggering	PduTriggering	*	aggr	One PduTriggering is defined for exactly one channel. Channels may have assigned an arbitrary number of I-Pdu triggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings must be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild

Table 8.4: PhysicalChannel

Class	≪atpVariation≫ **EthernetCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet-specific cluster attributes. **Tags:** atp.recommendedPackage=CommunicationClusters			
Base	ARObject, CollectableElement, CommunicationCluster, FibexElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
couplingPortConnection	CouplingPortConnection	*	aggr	Specification of connections between CouplingElements and EcuInstances. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=couplingPortConnection, variationPoint.shortLabel vh.latestBindingTime=postBuild
couplingPortSwitchoffDelay	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).
macMulticastGroup	MacMulticastGroup	*	aggr	MacMulticastGroup that is defined for the Subnet (EthernetCluster).

Table 8.5: EthernetCluster

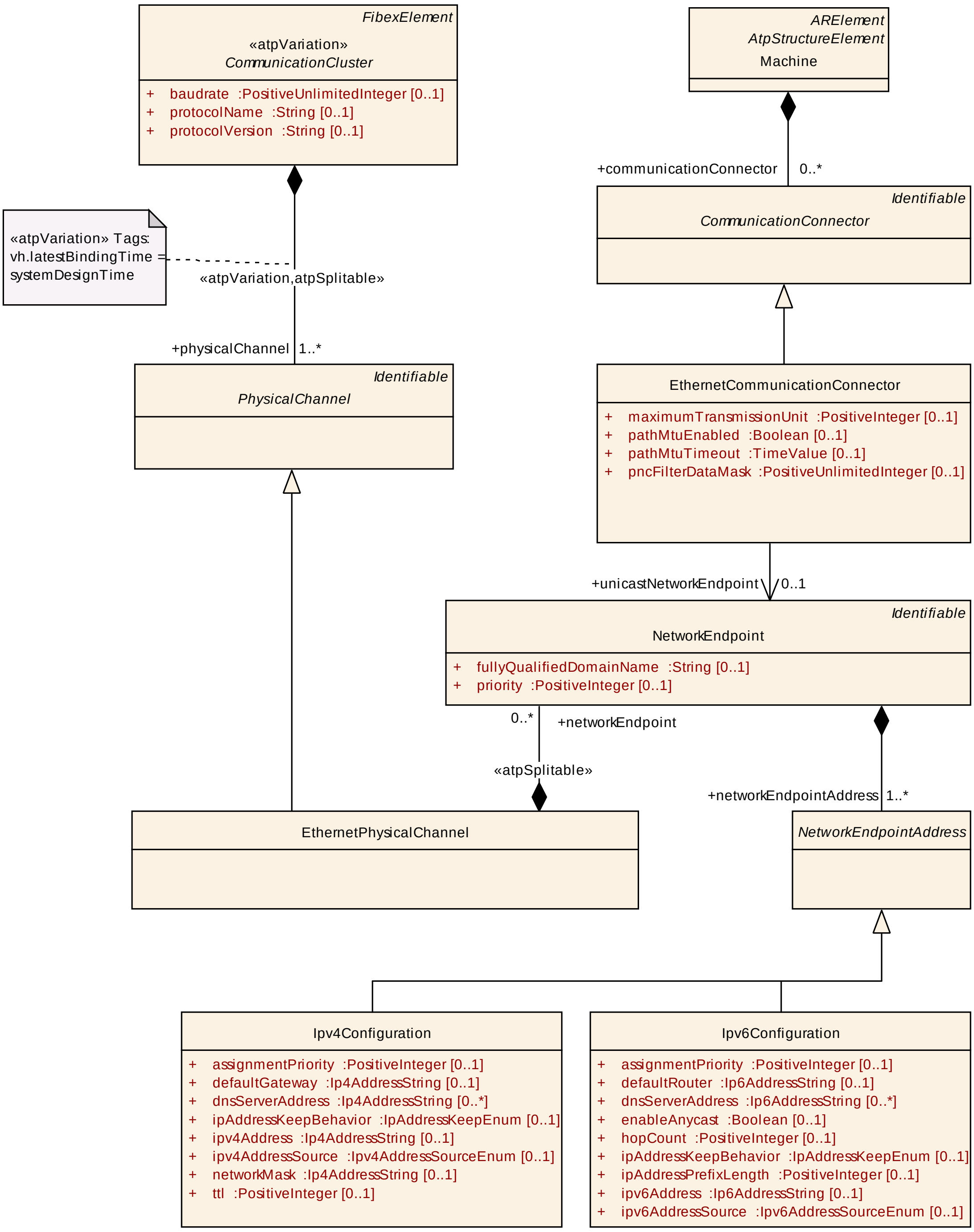

Figure 8.2: Network connection of a Machine

[constr_3320] Aggregation of `CommunicationConnector` by `Machine` ⌈ Meta-Class `Machine` shall only aggregate `EthernetCommunicationConnector`s 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 `Machine` 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).			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
fullyQualifiedDomainName	String	0..1	attr	Defines the fully qualified domain name (FQDN) e.g. some.example.host.
infrastructureServices	InfrastructureServices	0..1	aggr	Defines the network infrastructure services provided or consumed.
networkEndpointAddress	NetworkEndpointAddress	1..*	aggr	Definition of a Network Address. **Tags:** xml.namePlural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Priority of this Network-Endpoint.

Table 8.6: NetworkEndpoint

More precisely, the particular IP address is configured by means of the aggregation of Ipv4Configuration resp. 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] 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**
assignmentPriority	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.

defaultGateway	Ip4AddressString	0..1	attr	IP address of the default gateway.
dnsServerAddress	Ip4AddressString	*	attr	IP addresses of preconfigured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
ipAddressKeepBehavior	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 ipv4AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv4AddressSource	Ipv4AddressSourceEnum	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 8.7: 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 8.8: Ipv4AddressSourceEnum

[TPS_MANI_03053] 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::Ethernet Topology			
Note	Internet Protocol version 6 (IPv6) configuration.			
Base	ARObject, NetworkEndpointAddress			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
assignmentPriority	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.
dnsServerAddress	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)
ipAddressKeepBehavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipAddressPrefixLength	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 ipv6AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv6AddressSource	Ipv6AddressSourceEnum	0..1	attr	Defines how the node obtains its IP address.

Table 8.9: Ipv6Configuration

Enumeration	**Ipv6AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::Ethernet Topology
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
routerAdvertisement	IPv6 Stateless Autoconfiguration. **Tags:** atp.EnumerationValue=4

Table 8.10: 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 8.11: IpAddressKeepEnum

8.2 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::Deployment::Machine			
Note	Service Discovery configuration settings for the middleware transport layer. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.12: ServiceDiscoveryConfiguration

8.2.1 SOME/IP Service Discovery Configuration

[TPS_MANI_03064] SOME/IP Service Discovery message exchange configuration ⌈ ProvidedServiceInstances 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)*

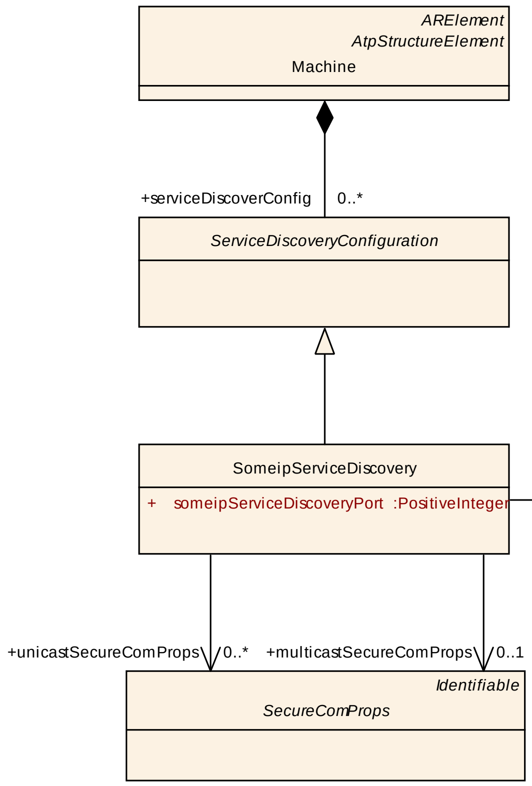

Figure 8.3: SOME/IP Service Discovery Configuration

Class	**SomeipServiceDiscovery**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	This meta-class represents a specialization of the generic service discovery for the SOME/IP case. **Tags:** atp.Status=draft			
Base	ARObject, ServiceDiscoveryConfiguration			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
multicastSdIpAddress	NetworkEndpoint	0..1	ref	This reference identifies the multicast IP address used for service discovery. **Tags:** atp.Status=draft

multicastSecureComProps	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
someipServiceDiscoveryPort	PositiveInteger	1	attr	This attribute represents the port number reserved for service discovery.
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 8.13: 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.

8.3 Hardware Resources

[TPS_MANI_03065] 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 [17].

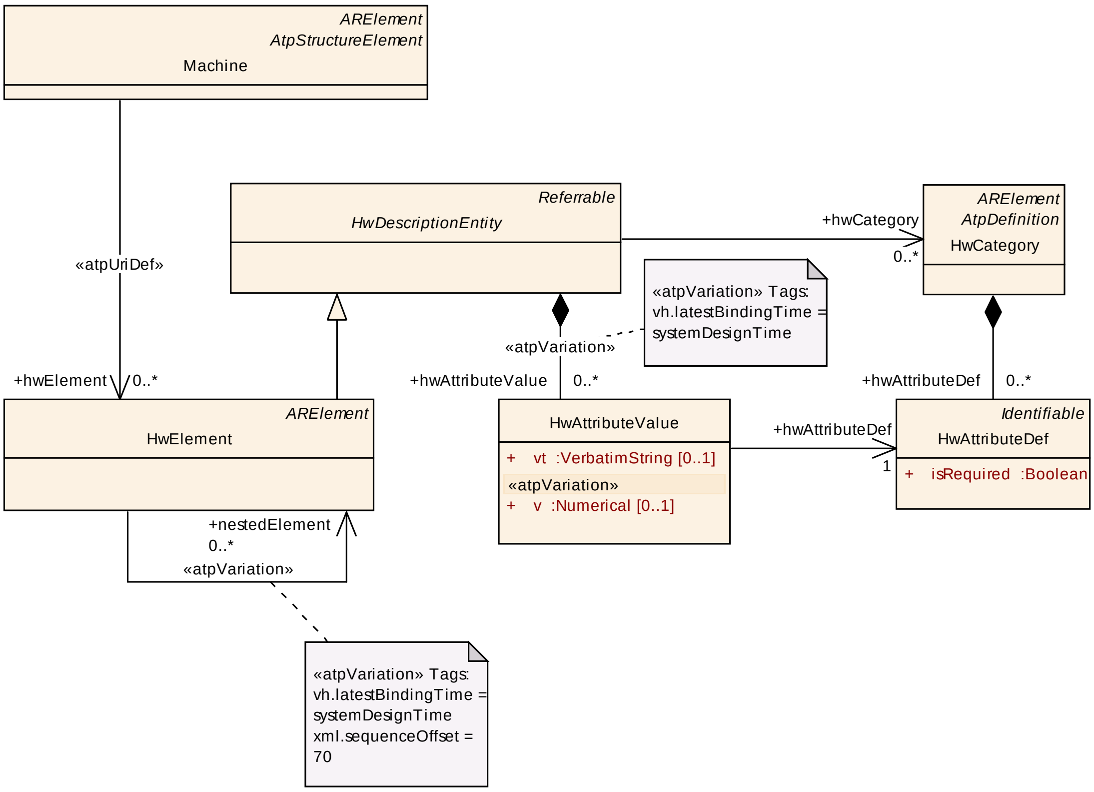

Figure 8.4: Description of hardware resources of the machine

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***
hwElementConnection	HwElementConnector	*	aggr	This represents one particular connection between two hardware elements. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=110
hwPinGroup	HwPinGroup	*	aggr	This aggregation is used to describe the connection facilities of a hardware element. Note that hardware element has no pins but only pingroups. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=90

nestedElement	HwElement	*	ref	This association is used to establish hierarchies of hw elements. Note that one particular HwElement can be target of this association only once. I.e. multiple instantiation of the same HwElement is not supported (at any hierarchy level). **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=70

Table 8.14: HwElement

Class	**HwDescriptionEntity (abstract)**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class represents the ability to describe a hardware entity.			
Base	ARObject, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hwAttributeValue	HwAttributeValue	*	aggr	This aggregation represents a particular hardware attribute value. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=50
hwCategory	HwCategory	*	ref	One of the associations representing one particular category of the hardware entity. **Tags:** xml.sequenceOffset=30
hwType	HwType	0..1	ref	This association is used to assign an optional HwType which contains the common attribute values for all occurences of this HwDescriptionEntity. Note that HwTypes can not be redefined and therefore shall not have a hwType reference.

Table 8.15: 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 HwAttributeValue.
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 8.16: 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, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hwAttribute Def	HwAttributeDef	*	aggr	This aggregation describes particular hardware attribute definition.

Table 8.17: 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 hwAttributeEnumerationLiterals specify the available literals.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hwAttribute Literal	HwAttributeLiter alDef	*	aggr	The available EnumerationLiterals of the Enumeration definition. Only applicable if the category of the HwAttributeDef 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 8.18: HwAttributeDef

8.4 Machine States

[TPS_MANI_03066] Description of machine states ⌈ With the machineModeMachine aggregation it is possible to define 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 ModeDeclarations and a designated initialMode. ⌋*(RS_MANI_00021)*

Please note that the startup of a Process may depend on Modes that are defined in the context of a Machine. The ModeDependentStartupConfig is described in chapter 6.2.

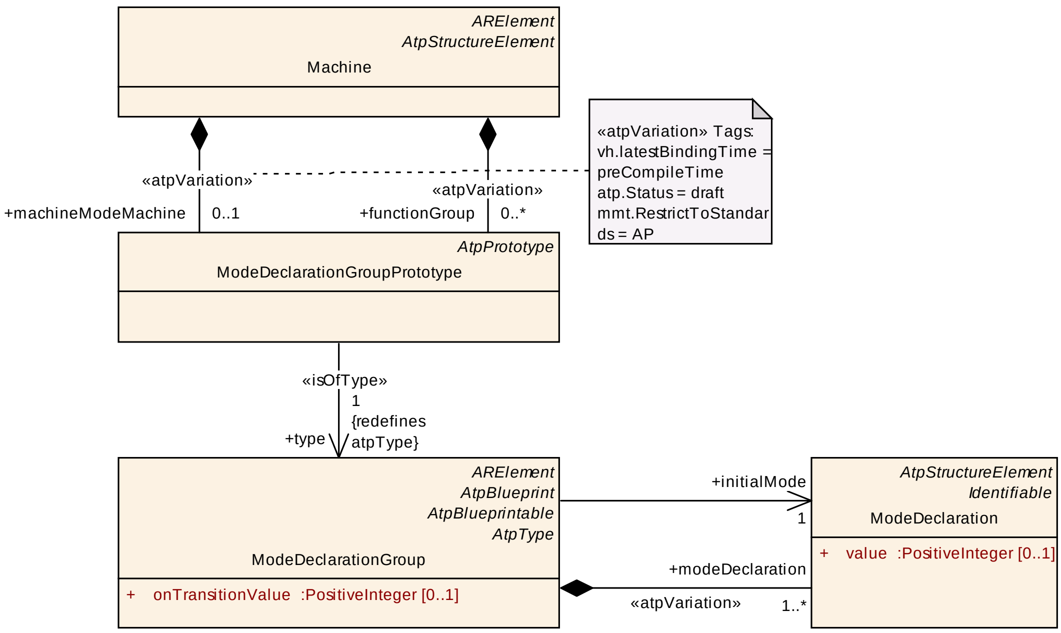

Figure 8.5: Configuration of Machine States

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.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
type	ModeDeclarationGroup	1	tref	The "collection of ModeDeclarations" (= ModeDeclarationGroup) supported by a component **Stereotypes:** isOfType

Table 8.19: ModeDeclarationGroupPrototype

Class	ModeDeclarationGroup			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.recommendedPackage=ModeDeclarationGroups			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, 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.
modeDeclaration	ModeDeclaration	1..*	aggr	The ModeDeclarations collected in this ModeDeclarationGroup. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
modeManagerErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user).
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup
modeUserErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager).
onTransitionValue	PositiveInteger	0..1	attr	The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses.

Table 8.20: 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.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this ModeDeclaration.

Table 8.21: ModeDeclaration

8.5 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 ModeDependentStartupConfig refers to the function group state with the functionGroup reference.

The usage of Function Groups is described in more detail in [15].

[TPS_MANI_03145] 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 ModeDeclarations and a designated initialMode. ⌋*()*

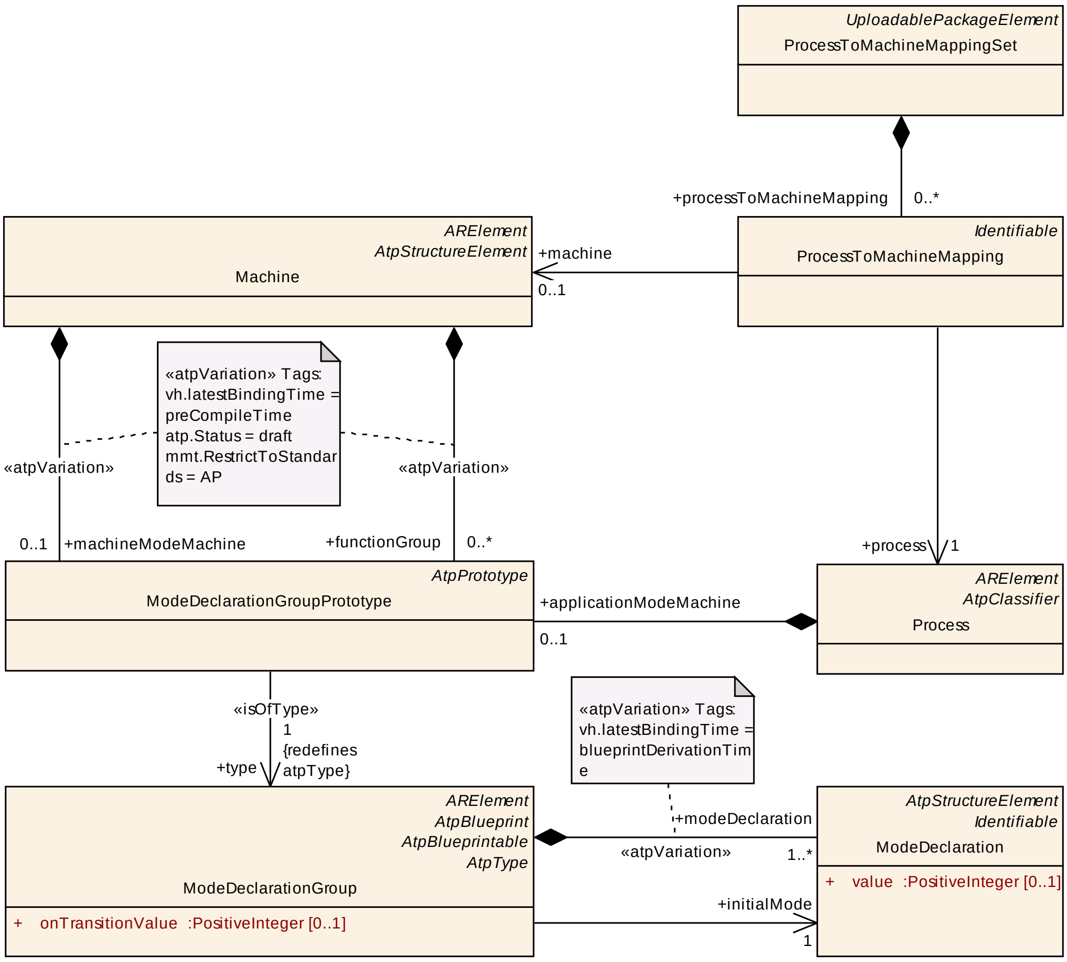

Figure 8.6: Configuration of Function Groups

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.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
type	ModeDeclarationGroup	1	tref	The "collection of ModeDeclarations" (= ModeDeclarationGroup) supported by a component **Stereotypes:** isOfType

Table 8.22: ModeDeclarationGroupPrototype

Class	**ModeDeclarationGroup**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.recommendedPackage=ModeDeclarationGroups			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, 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.
modeDeclaration	ModeDeclaration	1..*	aggr	The ModeDeclarations collected in this ModeDeclarationGroup. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
modeManagerErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user).
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup
modeUserErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager).
onTransitionValue	PositiveInteger	0..1	attr	The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses.

Table 8.23: 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.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this ModeDeclaration.

Table 8.24: ModeDeclaration

8.6 State Timeouts

[TPS_MANI_03146] 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. ⌋*()*

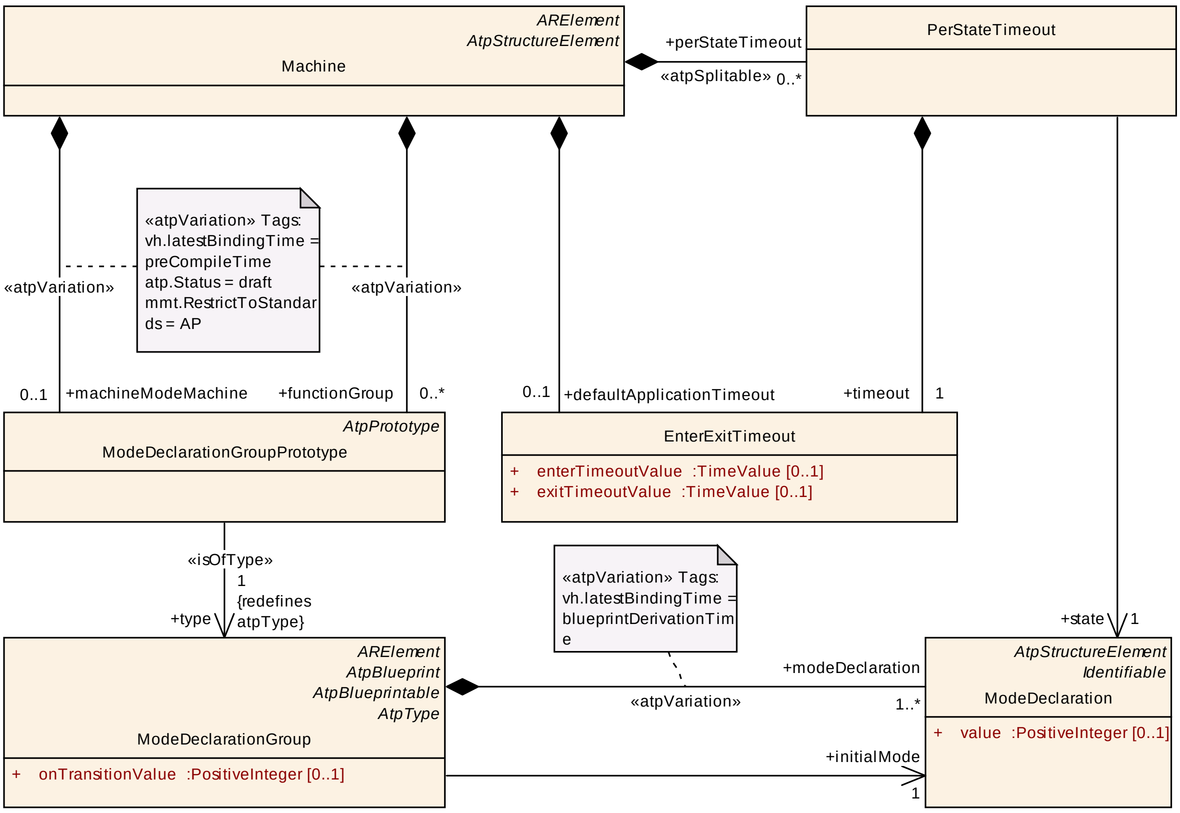

Figure 8.7: Configuration of timeouts for selected machine states and function group states

Class	**PerStateTimeout**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Machine			
Note	This meta-class represents the ability to specify a state-specific timeout. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
state	ModeDeclaration	1	ref	Ths 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 8.25: PerStateTimeout

Class	**EnterExitTimeout**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Machine			
Note	This meta-class represents the ability to specify a pair of timeouts, one for entering, and one for exiting. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
enterTimeoutValue	TimeValue	0..1	attr	This attribute represents the value of the enter timeout in seconds.
exitTimeoutValue	TimeValue	0..1	attr	This attribute represents the value of the exit timeout in seconds.

Table 8.26: EnterExitTimeout

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 [15].

8.7 Process To Machine Mapping

[TPS_MANI_03147] Mapping of a `Process` to a `Machine` ⌈ The meta-class `ProcessToMachineMapping` provides the ability to map a `Process` to a `Machine`. ⌋ *()*

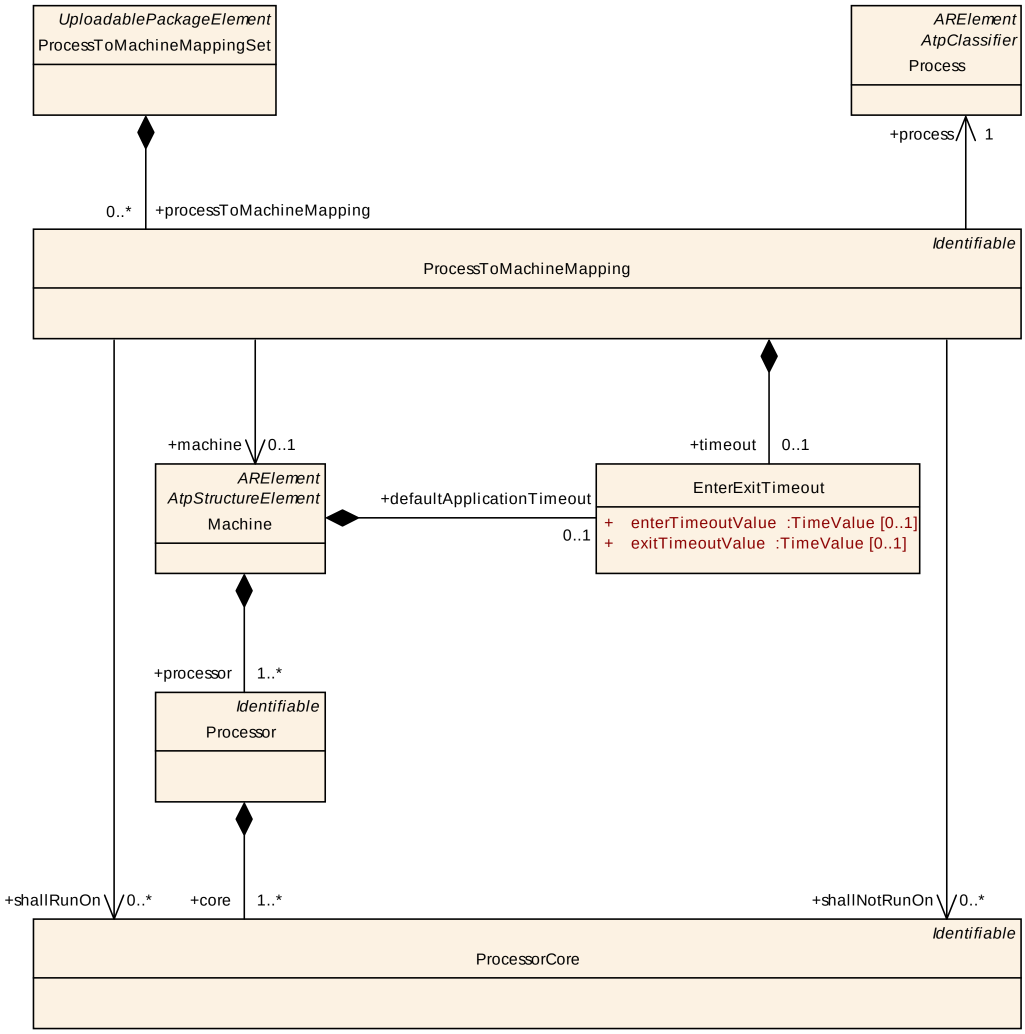

Figure 8.8: Mapping of a `Process` to a `Machine`

Class	**ProcessToMachineMappingSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Machine			
Note	This meta-class acts as a bucket for collecting ProcessToMachineMappings. **Tags:** atp.Status=draft; atp.recommendedPackage=ProcessToMachineMappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
processToMachineMapping	ProcessToMachineMapping	*	aggr	This represents the collection of ProcessToMachineMappings of the enclosing ProcessToMachineMappingSet. **Tags:** atp.Status=draft

Table 8.27: ProcessToMachineMappingSet

Class	**ProcessToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Machine			
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.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
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 8.28: ProcessToMachineMapping

[TPS_MANI_03148] 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] 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`. ⌋*()*

Class	**EnterExitTimeout**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Machine			
Note	This meta-class represents the ability to specify a pair of timeouts, one for entering, and one for exiting. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
enterTimeoutValue	TimeValue	0..1	attr	This attribute represents the value of the enter timeout in seconds.
exitTimeoutValue	TimeValue	0..1	attr	This attribute represents the value of the exit timeout in seconds.

Table 8.29: EnterExitTimeout

[TPS_MANI_03149] 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] 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] 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] 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. ⌋*()*

8.8 Adaptive Autosar Module and Platform Configuration

The configuration settings for individual Adaptive Autosar modules are covered by specializations of the abstract class AdaptiveModuleInstantiation.

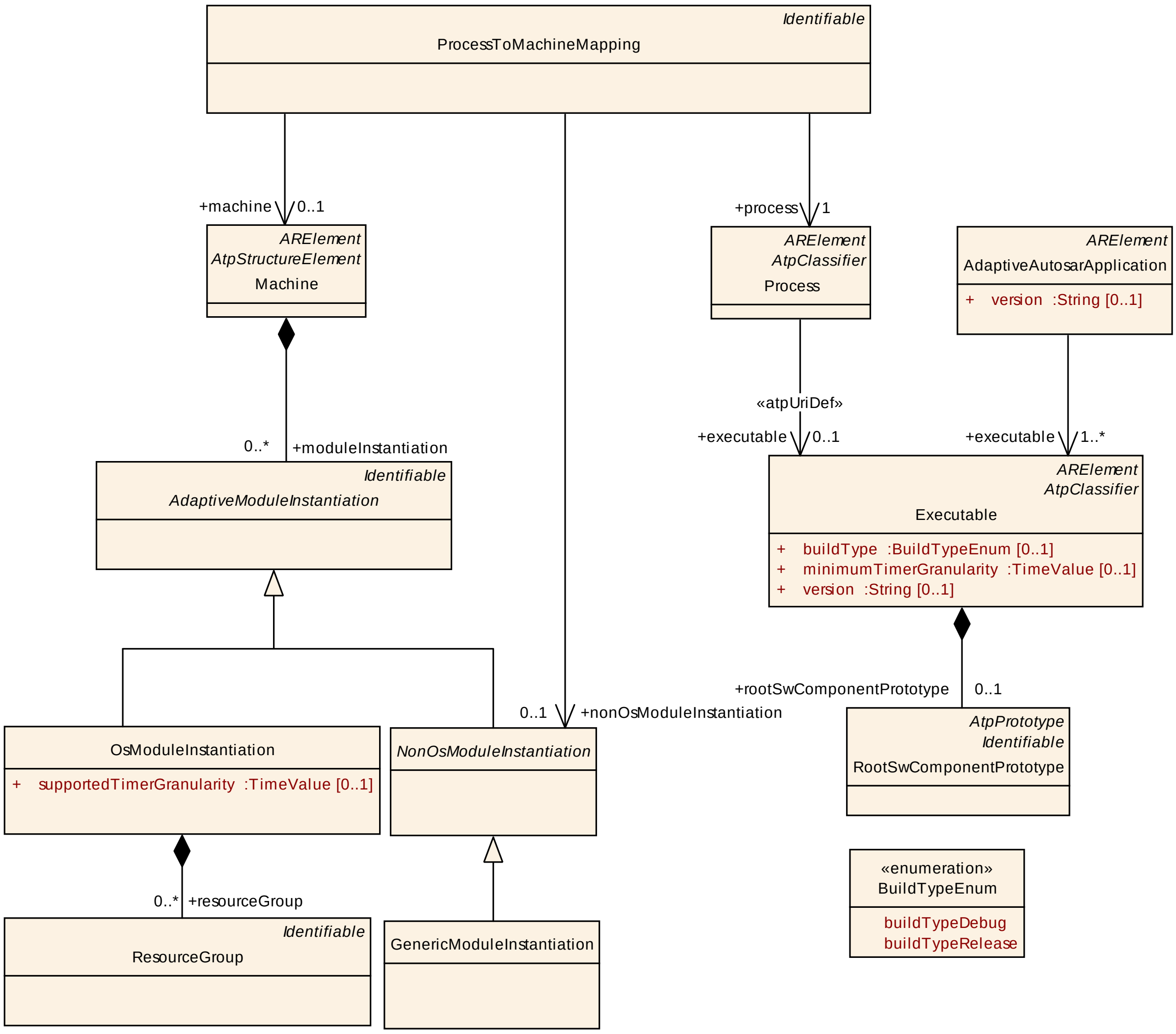

Figure 8.9: Adaptive Autosar Module Configuration

Class	**AdaptiveModuleInstantiation (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::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.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 8.30: AdaptiveModuleInstantiation

Each Adaptive Autosar module other than OS can be assigned to a `Process` with the `ProcessToMachineMapping`.

[constr_1490] 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 [TPS_MANI_01009]). ⌋*()*

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 [5].

[TPS_MANI_03096] 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::Deployment::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.Status=draft			
Base	ARObject, AdaptiveModuleInstantiation, Identifiable, MultilanguageReferrable, Non OsModuleInstantiation, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.31: GenericModuleInstantiation

8.8.1 OS Module configuration

[TPS_MANI_03098] 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::Deployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the OS configuration on a specific machine. **Tags:** 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
supportedTimerGranularity	TimeValue	0..1	attr	This attribute describes the supported timer granularity (TimeValue of one tick). **Tags:** atp.Status=draft

Table 8.32: OsModuleInstantiation

Class	**ResourceGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::AdaptiveModuleImplementation			
Note	This meta-class represents a resource group. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 8.33: ResourceGroup

9 System Design

9.1 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.

9.2 Specification of 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.

[constr_3366] System category for a system description with Adaptive Platform components ⌈ The System element that contains SwComponentPrototypes of AdaptiveApplicationSwComponentType nested inside the CompositionSwComponentType that is referenced by the RootSwCompositionPrototype shall have the category SOFTWARE_COMPONENT_SYSTEM_DESIGN_DESCRIPTION. ⌋*()*

[TPS_MANI_03110] Allowed components in system description with category category SOFTWARE_COMPONENT_SYSTEM_DESIGN_DESCRIPTION. ⌈ SwComponentPrototypes nested inside the CompositionSwComponentType that is referenced by the RootSwCompositionPrototype of a System with category SOFTWARE_COMPONENT_SYSTEM_DESIGN_DESCRIPTION are allowed to be of any SwComponentType that is supported by Classic or by Adaptive Autosar. ⌋*(RS_MANI_00026)*

Class	**System**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints. The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology. **Tags:** atp.recommendedPackage=Systems			
Base	ARElement, ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Collectable Element, 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 FibexElements can be optional. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
rootSoftwareComposition	RootSwCompositionPrototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=systemDesignTime
systemVersion	RevisionLabelString	1	attr	Version number of the System Description.

Table 9.1: System

Class	RootSwCompositionPrototype			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within a given System. According to the use case of the System, this may for example be the a more or less complete VFB description, the software of a System Extract or the software of a flat ECU Extract with only atomic SWCs. Therefore the RootSwComposition will only occasionally contain all atomic software components that are used in a complete VFB System. The OEM is primarily interested in the required functionality and the interfaces defining the integration of the Software Component into the System. The internal structure of such a component contains often substantial intellectual property of a supplier. Therefore a top-level software composition will often contain empty compositions which represent subsystems. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including PortPrototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Type	Mul.	Kind	Note
softwareComposition	CompositionSwComponentType	1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system **Stereotypes:** isOfType

Table 9.2: 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 Design model 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] Usage of AssemblySwConnectors in the System Design model ⌈ In the System Design model 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)*

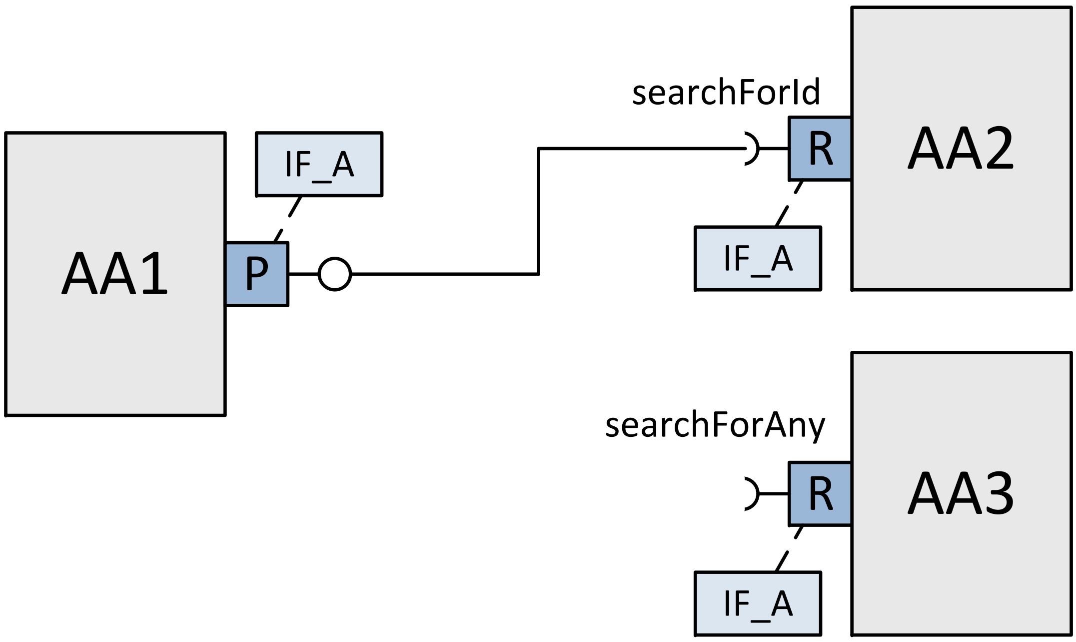

Figure 9.1: Example for Assembly connectors in System Design model

9.3 Modeling of service oriented communication between Classic and Adaptive platform

AUTOSAR classic platform does not support ServiceInterfaces 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 ClientServerInterfaces, SenderReceiverInterfaces or TriggerInterfaces 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 PortInterfaces 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, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

interfaceMapping	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 9.3: 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 SenderReceiverInterface to an Event in a ServiceInterface. **Tags:** atp.Status=draft
fieldMapping	FieldMapping	*	aggr	Mapping of a Field 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
fireAndForgetMapping	FireAndForgetMapping	*	aggr	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
methodMapping	MethodMapping	*	aggr	Mapping of a ClientServerOperation in a ClientServerInterface to a Method in a ServiceInterface. **Tags:** atp.Status=draft

Table 9.4: InterfaceMapping

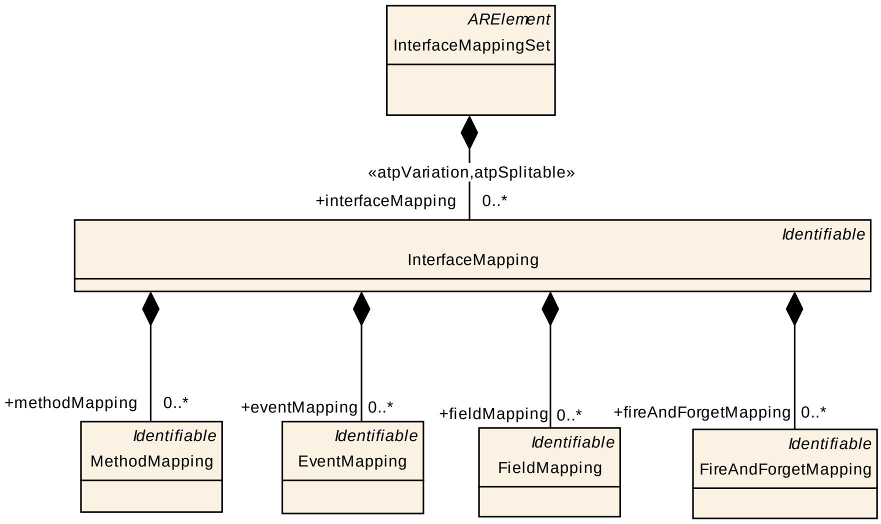

Figure 9.2: InterfaceMapping Overview

[constr_3370] 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 9.3 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.

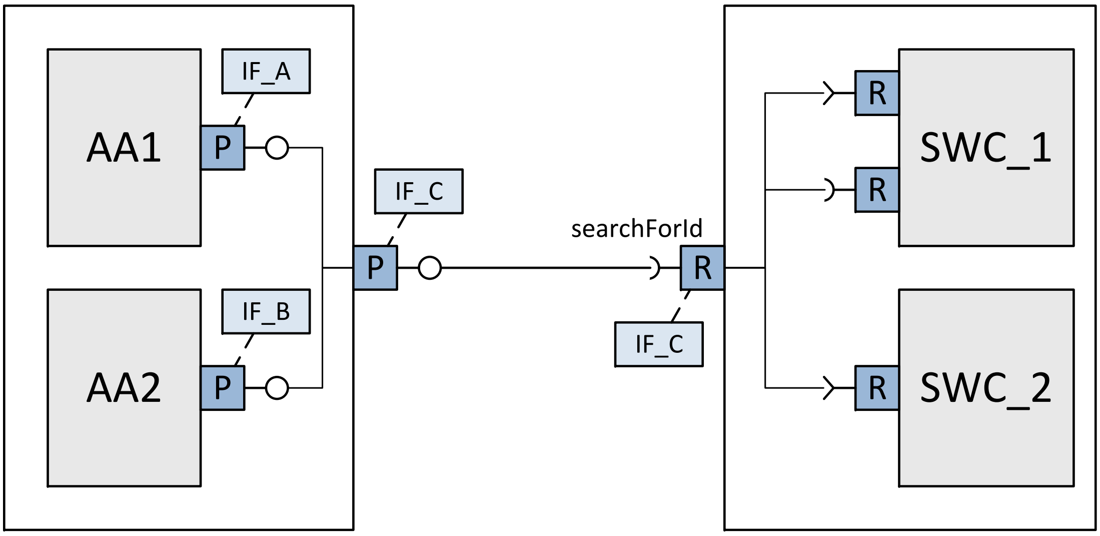

Figure 9.3: Example for a modeling of Service Oriented communication between Adaptive Applications and Software Components of the Classic Platform

9.3.1 MethodMapping

[TPS_MANI_03111] 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)*

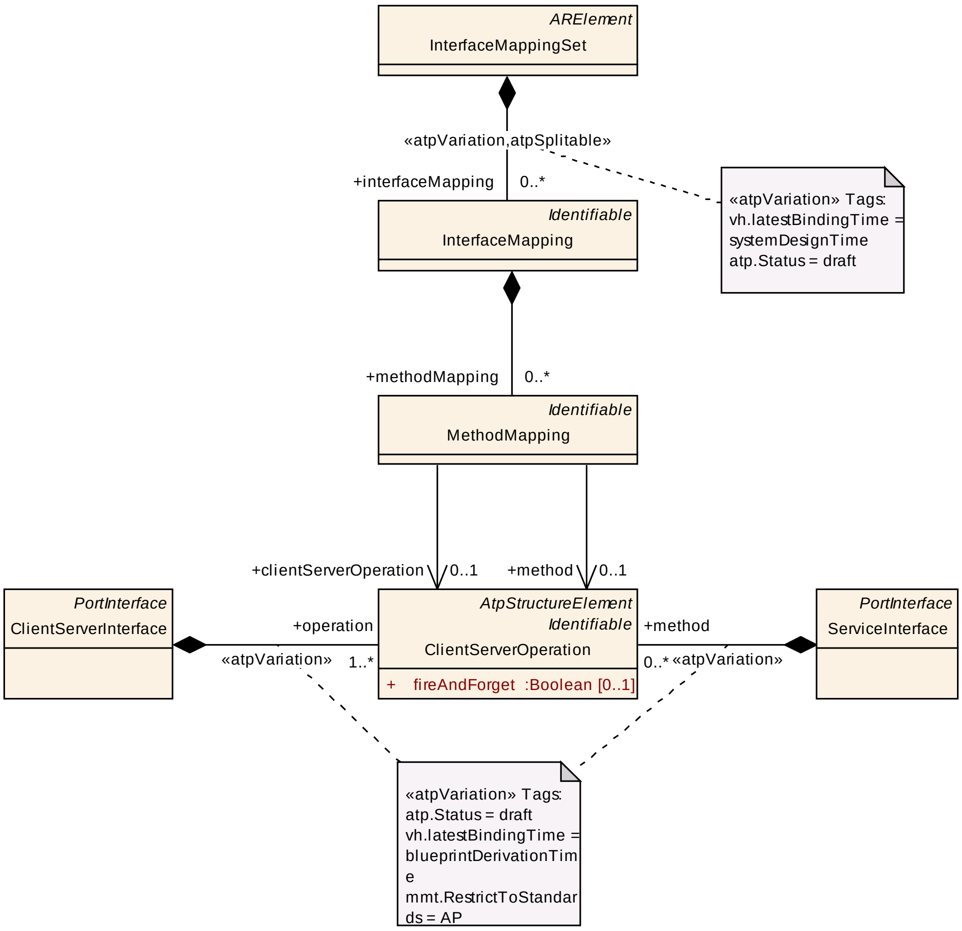

Figure 9.4: Mapping of a Method to a ClientServerOperation

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***
clientServerOperation	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 ServiceInterface. **Tags:** atp.Status=draft

Table 9.5: MethodMapping

9.3.2 EventMapping

[TPS_MANI_03112] 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)*

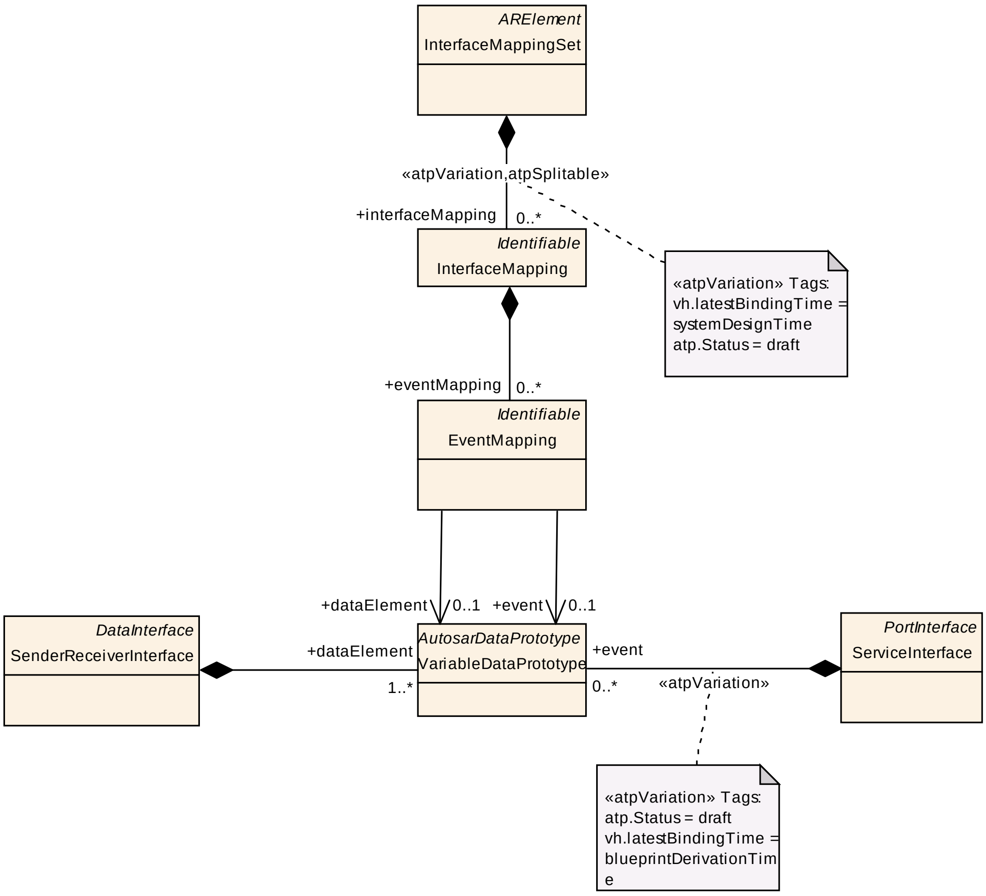

Figure 9.5: Mapping between an event and a dataElement

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

event	VariableDataPrototype	0..1	ref	Reference to an Event that is located in a ServiceInterface. **Tags:** atp.Status=draft

Table 9.6: EventMapping

9.3.3 FieldMapping

[TPS_MANI_03113] Mapping between a `field` and elements of Classic Platform `PortInterfaces` ⌈ The mapping between a `field` located in a `ServiceInterface` and elements of Classic Platform `PortInterface`s 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 `operation`s that are located in a `ClientServerInterface`. ⌋*(RS_MANI_00026)*

[constr_3367] `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] `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] `FieldMapping.setterOperation` reference ⌈ The `FieldMapping` shall only contain the `setterOperation` reference if the `hasSetter` attribute in the referenced `field` is set to true. ⌋*()*

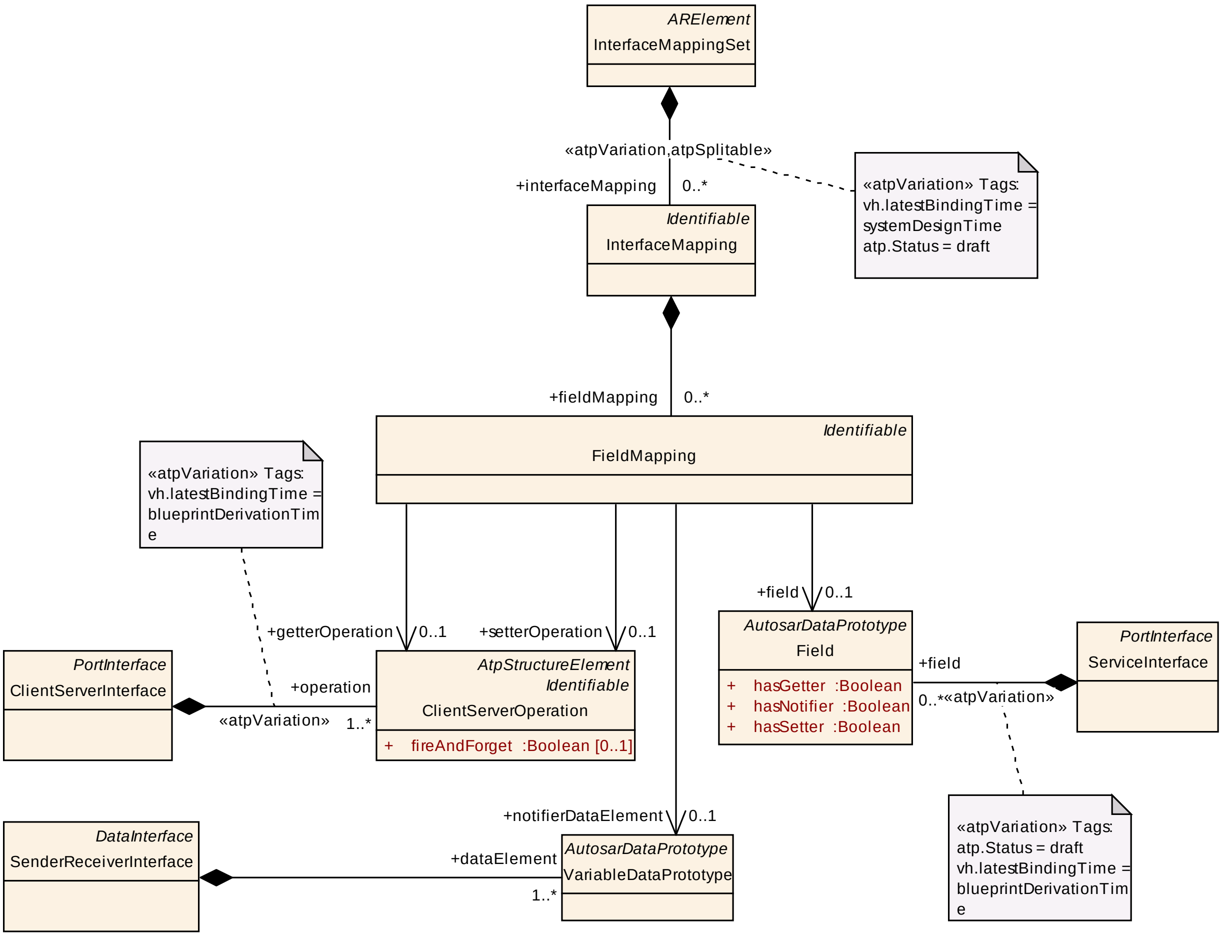

Figure 9.6: Mapping between a field and elements of Classic Platform PortInterfaces

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
notifierDataElement	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that represents the notifier in the Field. **Tags:** atp.Status=draft

setterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the setter Method in the Field. **Tags:** atp.Status=draft

Table 9.7: FieldMapping

9.3.4 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] Mapping between a fire and forget `method` and elements of Classic Platform `PortInterfaces` ⌈ 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] 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] 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.⌋()

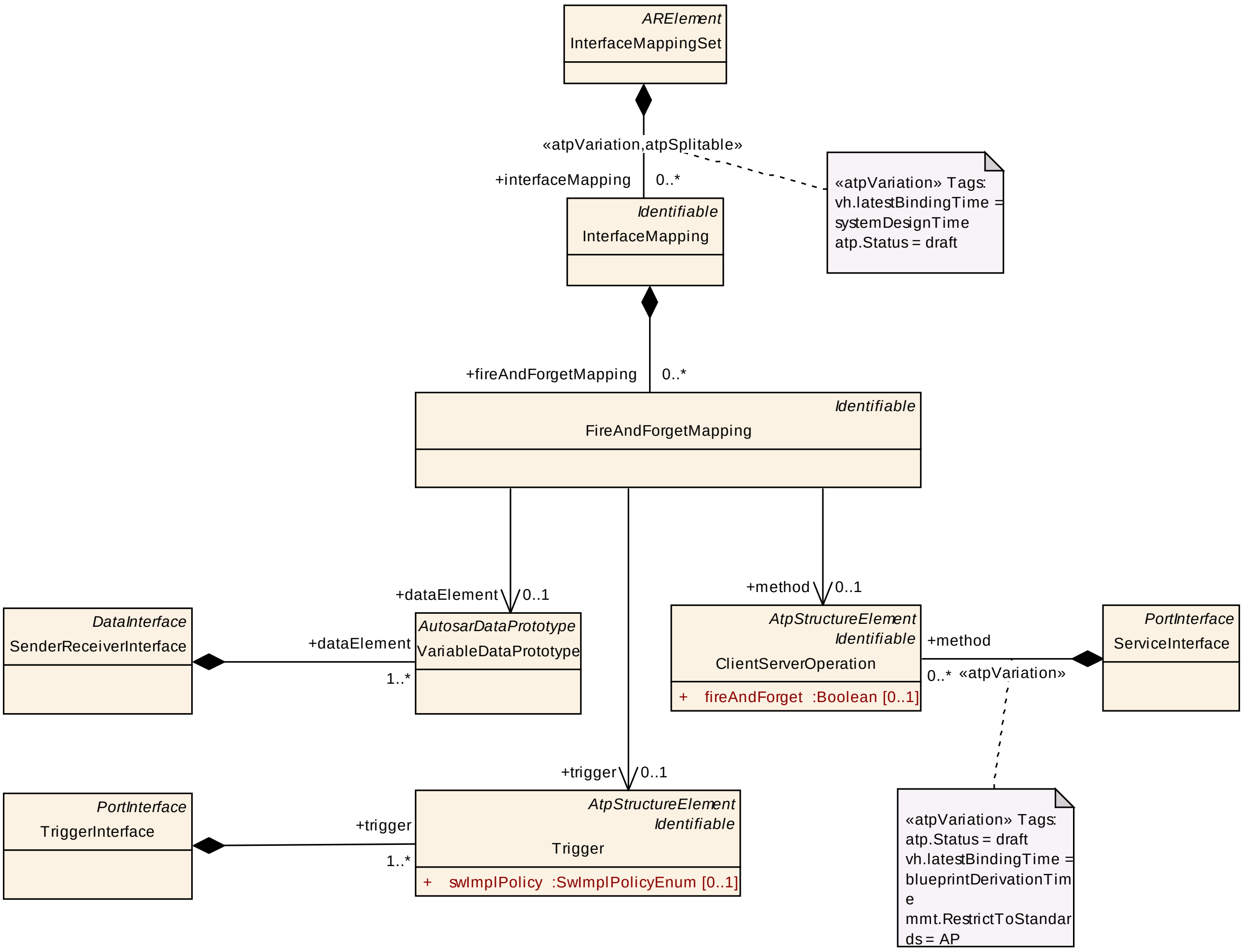

Figure 9.7: Mapping between a fire and forget method and elements of Classic Platform PortInterfaces

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			
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 9.8: FireAndForgetMapping

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 `ServiceInterface`s 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 s-ketched 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 `ISignal`s 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 `Pdu`s and `ISignal`s.

This chapter introduces a modeling that creates a bridge between the service-oriented communication based on `ServiceInterface`s of the *AUTOSAR adaptive platform* and the signal-based communication involving the definition of `Pdu`s and `ISignal`s 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 plat-*

form System Description would also contain a Pdu or Signal Gateway configuration between the Ethernet and the CAN bus.

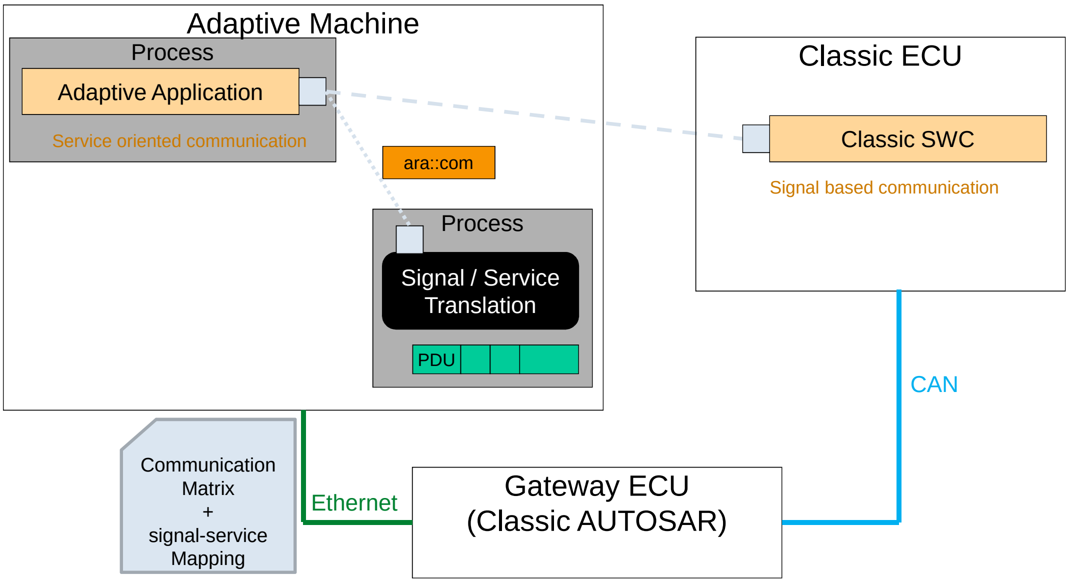

Figure 10.1: SignalToService translation in Application on Adaptive Machine

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.

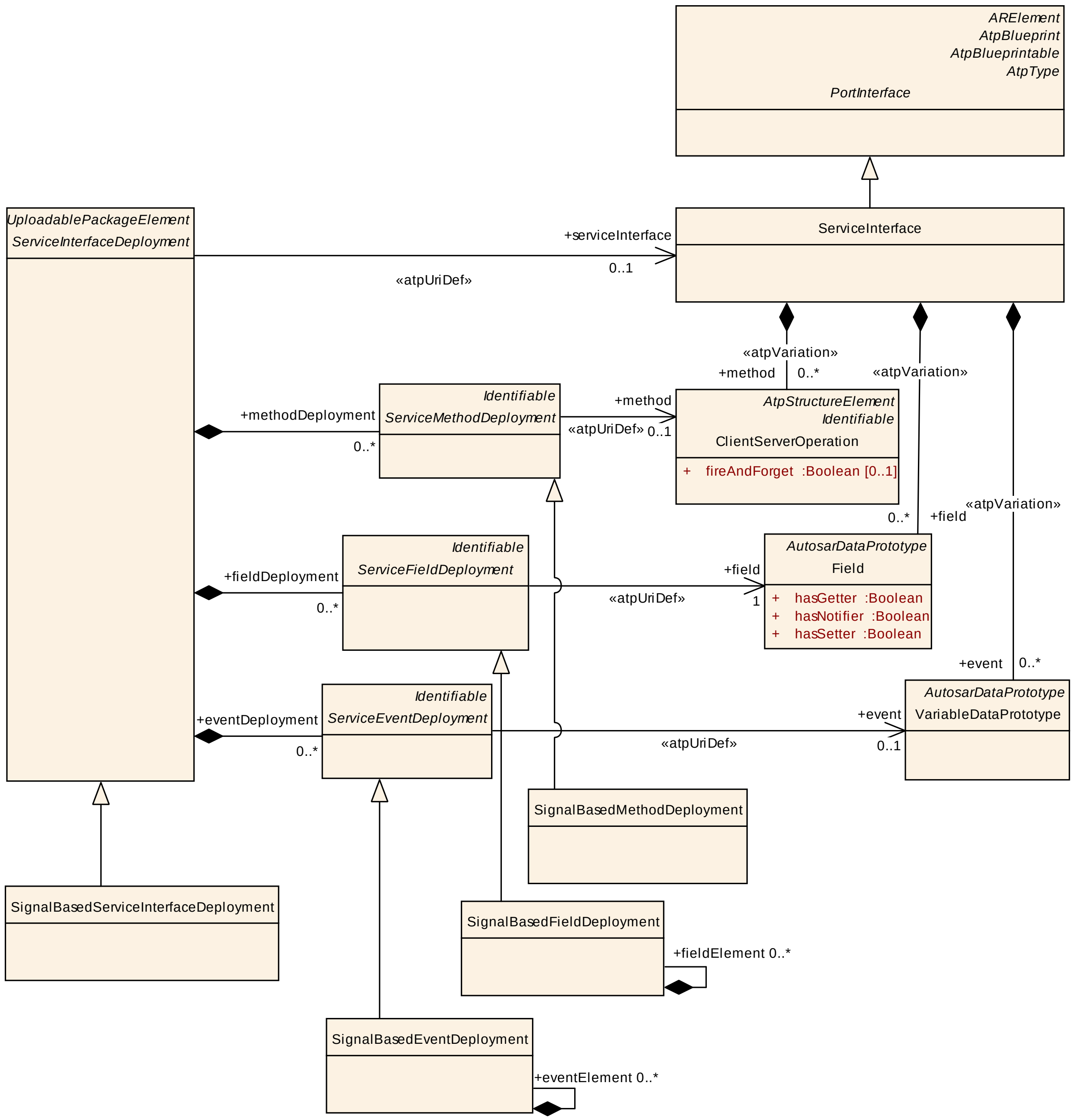

Figure 10.2: Signal-based deployment of ServiceInterface

[TPS_MANI_03120] 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 `ServiceInterfaceDeployment`s that are described in section 7.1, 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::Deployment::ServiceInterface Deployment			
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, PackageableElement, Referrable, ServiceInterfaceDeployment, UploadablePackage Element			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 10.1: SignalBasedServiceInterfaceDeployment

The meta-class `SignalBasedEventDeployment` allows to flatten the structure of the referenced `VariableDataPrototype` with the `eventElement` aggregation, as shown in figure Figure 10.3, where all primitive elements that are defined inside of the Event that is typed by a Structure are modeled as `eventElement`s. This allows for the later mapping of these `eventElement`s to individual signals.

[TPS_MANI_03121] Signal-based `VariableDataPrototype` binding ⌈ The `SignalBasedEventDeployment` meta-class provides the ability to map a `VariableDataPrototype` that is referenced in the role `event` to one or several `ISignal`s. ⌋ *(RS_MANI_00029)*

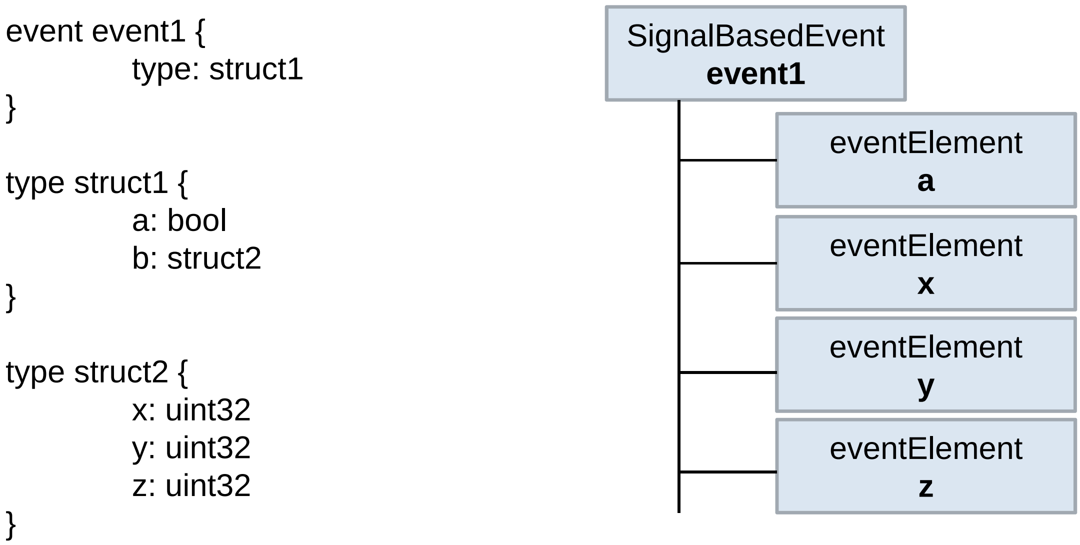

Figure 10.3: Usage of SignalBasedEvent

Class	**SignalBasedEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	This element needs to be defined if the event needs to be transported in a signal-based way over a communication channel. If the datatype of the event is composite then the hierarchy and all primitive dataelements need to be described with the aggregated eventElements since every single eventElement will be transported in an individual signal over the communication medium. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceEvent Deployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventElement	SignalBasedEventDeployment	*	aggr	In case that the datatype of the event is composite all primitive elements of the datatype need to be described since every single one will be transported in an individual Signal over the communication medium. **Tags:** atp.Status=draft

Table 10.2: SignalBasedEventDeployment

[TPS_MANI_03122] Signal-based `Field` binding ⌈ The `SignalBasedFieldDeployment` meta-class provides the ability to map a `Field` that is referenced in the role `field` to one or several `ISignal`s. ⌋*(RS_MANI_00029)*

Class	**SignalBasedFieldDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	This element needs to be defined if the field needs to be transported in a signal-based way over a communication channel. If the datatype of the field is composite and a notifier is defined in the field then the datatype hierarchy and all primitive dataelements need to be described with the aggregated fieldElements since every single fieldElement will be transported in an individual signal over the communication medium. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceFieldDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
fieldElement	SignalBasedFieldDeployment	*	aggr	In case that the datatype of the field is composite and a notifier is defined for the field all primitive elements of the datatype need to be described since every single one will be transported in an individual Signal over the communication medium. **Tags:** atp.Status=draft

Table 10.3: SignalBasedFieldDeployment

If the attribute `hasNotifier` in the referenced `Field` is set to true, the `SignalBasedFieldDeployment` needs to be handled in the same way as the `Signal-`

BasedEventDeployment, i.e. all primitive elements that are defined inside of the Notifier that is typed by a Structure are modeled as fieldElements.

If the attribute hasNotifier in the referenced Field is set to false, no field-Element need to be defined. The reason is that a ClientServerOperation in *AUTOSAR classic platform* is always mapped to a single Call-Signal and a single Return-Signal, and a mapping of individual arguments to Signals is not supported. If the Field has only the getter and/or setter method, all necessary information to describe the Signal-to-Service mapping is already available with the SignalBased-FieldDeployment .

For the same reason, the SignalBasedMethodDeployment does not contain any aggregations.

[TPS_MANI_03123] Signal-based ClientServerOperation binding ⌈ The SignalBasedMethodDeployment meta-class provides the ability to map a ClientServerOperation that is referenced in the role method to one or several ISignals. ⌋*(RS_MANI_00029)*

Class	**SignalBasedMethodDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInterface Deployment			
Note	This element needs to be defined if the method needs to be transported in a signal-based way over a communication channel. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceMethod Deployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 10.4: SignalBasedMethodDeployment

10.3 Signal-To-Service Mapping

This chapter describes the mapping of ServiceInterface elements of a specific AdaptivePlatformServiceInstance defined in the context of a Process to ISignalTriggerings. The prerequisite is the definition of the SignalBasedServiceInterfaceDeployment with all necessary Signal-based methodDeployments, fieldDeployments and eventDeployments.

Class	ServiceInstanceToSignalMappingSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceElementToSignalMapping			
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 ServiceInstance to a number of ISignalTriggerings, **Tags:** atp.Status=draft

Table 10.5: ServiceInstanceToSignalMappingSet

Class	ServiceInstanceToSignalMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceElementToSignalMapping			
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
serviceInstance	ServiceInstanceToPortPrototypeMapping	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 10.6: ServiceInstanceToSignalMapping

The `ServiceInstanceToSignalMapping` references a `ServiceInstanceToPortPrototypeMapping` and thereby defines the `AdaptivePlatformServiceInstance` executed in a `Process` of 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

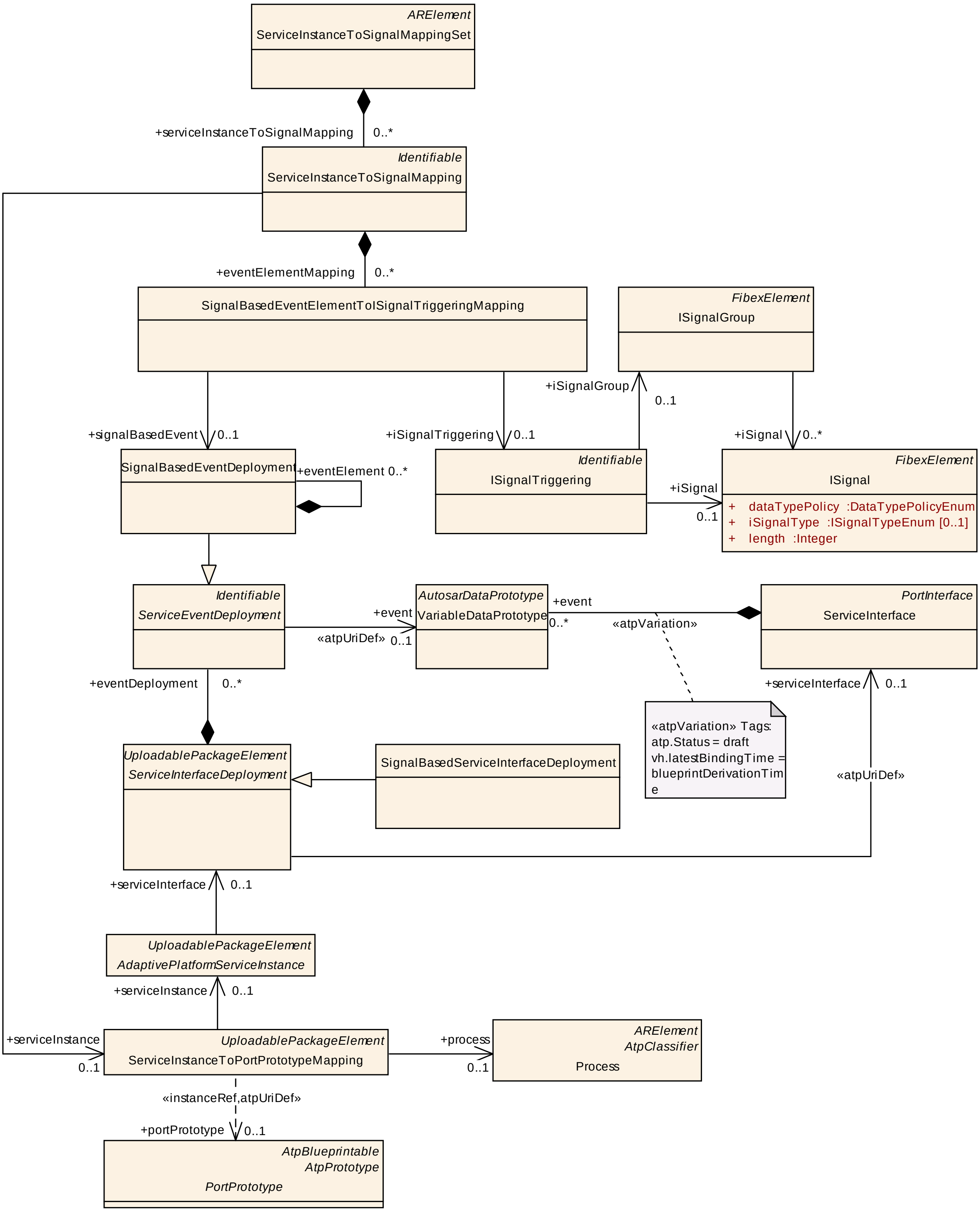

Figure 10.4: Mapping of Event elements to ISignals

[TPS_MANI_03124] SignalBasedEventDeployment to ISignalTriggering mapping ⌈ The SignalBasedEventElementToISignalTriggeringMapping meta-class provides the ability to map a SignalBasedEventDeployment that is referenced in the role signalBasedEvent to one ISignalTriggering of the ISignal or ISignalGroup. ⌋*(RS_MANI_00029)*

In the example sketched in Figure 10.3, one SignalBasedEventElementToISignalTriggeringMapping would map the SignalBasedEventDeployment *event1* to an ISignalTriggering of an ISignalGroup. Another SignalBasedEventElementToISignalTriggeringMapping would map the eventElement *a* to an ISignalTriggering of an ISignal contained in the ISignalGroup. Finally, one more SignalBasedEventElementToISignalTriggeringMappings would map the eventElements *x*, *y* and *z* to additional ISignalTriggerings of individual ISignals located in the same ISignalGroup.

Class	**SignalBasedEventElementToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceElementToSignalMapping			
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***
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
signalBasedEvent	SignalBasedEventDeployment	0..1	ref	Reference to an Event or an element inside of the Event that will be mapped to an ISignalTriggering for signal-based transport over a communication channel. **Tags:** atp.Status=draft

Table 10.7: SignalBasedEventElementToISignalTriggeringMapping

10.3.2 SignalBasedField Mapping

[TPS_MANI_03126] SignalBasedFieldDeployment to ISignalTriggerings mapping ⌈ The SignalBasedFieldToISignalTriggeringMapping meta-class provides the ability to map a SignalBasedFieldDeployment that is referenced in the role signalBasedField

- to one ISignalTriggering for the ISignalGroup representing the Notifier or

- to one ISignalTriggering for the ISignal representing the Notifier element (fieldElement) or
- to one ISignalTriggering for the ISignal representing the Getter-Call or
- to one ISignalTriggering for the ISignal representing the Getter-Return or
- to one ISignalTriggering for the ISignal representing the Setter-Call or
- to one ISignalTriggering for the ISignal representing the Setter-Return or

⌋*(RS_MANI_00029)*

It means that several SignalBasedFieldToISignalTriggeringMappings are necessary to map a SignalBasedFieldDeployment to a number of ISignalTriggerings.

[constr_3377] Restriction of ISignalTriggering references in SignalBasedFieldToISignalTriggeringMapping ⌈ For any given SignalBasedFieldToISignalTriggeringMapping, only a single reference to an ISignalTriggering shall exist. ⌋*()*

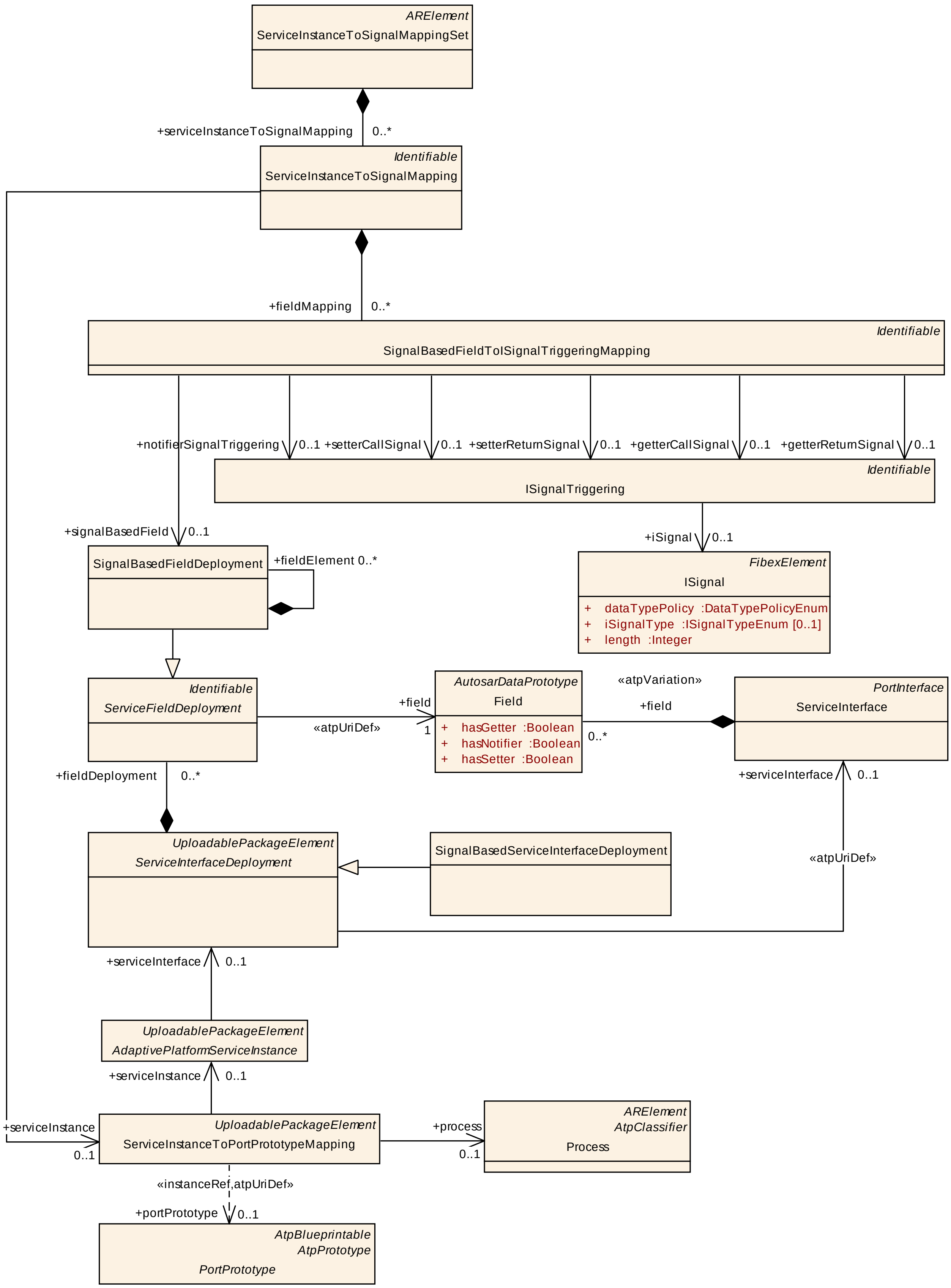

Figure 10.5: Mapping of Fields to ISignals

Class	**SignalBasedFieldToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceElementToSignalMapping			
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**
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
signalBasedField	SignalBasedFieldDeployment	0..1	ref	Reference to an field or an element inside of the field that will be mapped to an ISignalTriggering for signal-based transport over a communication channel. **Tags:** atp.Status=draft

Table 10.8: SignalBasedFieldToISignalTriggeringMapping

10.3.3 SignalBasedMethod Mapping

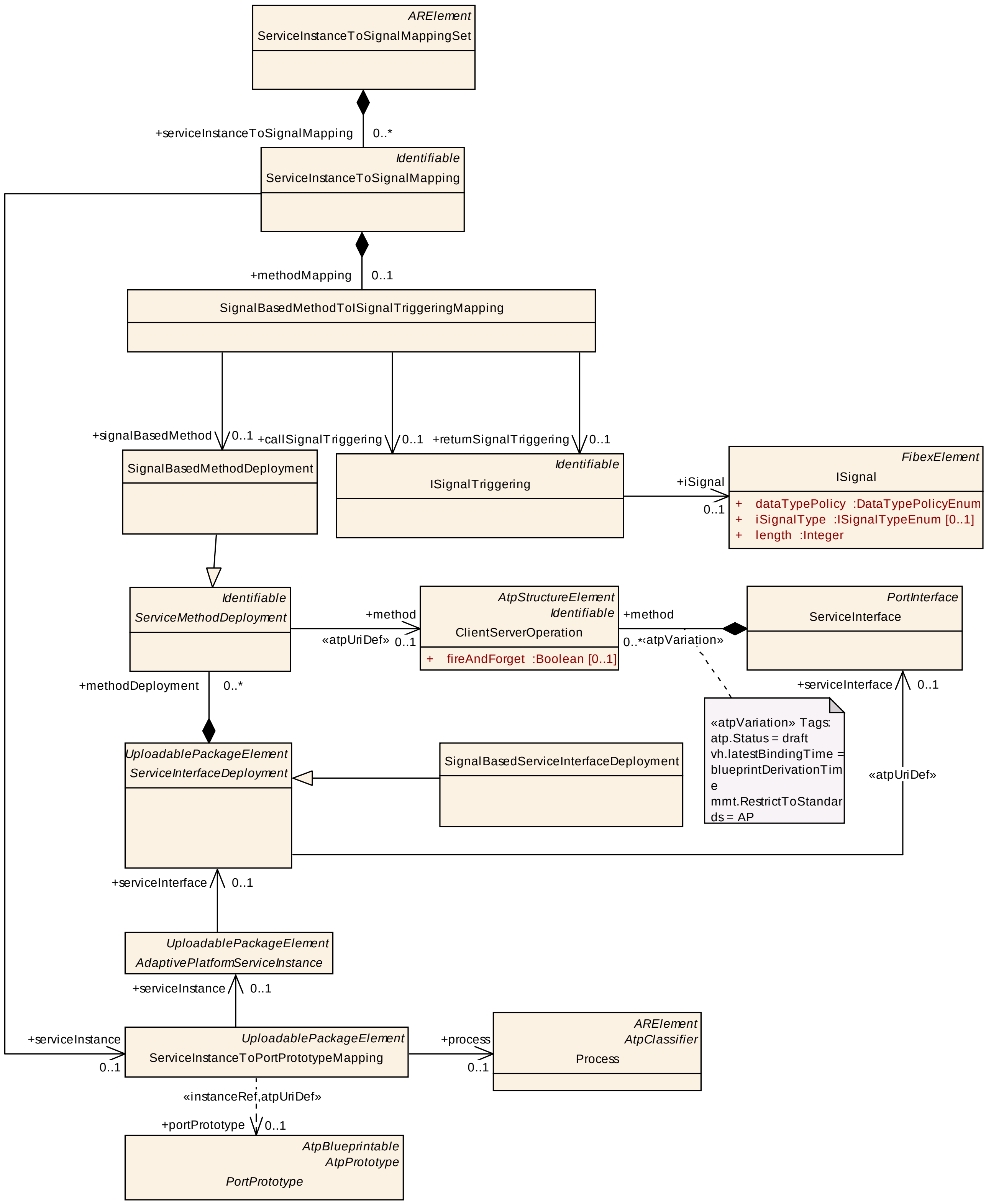

Figure 10.6: Mapping of Methods to ISignals

[TPS_MANI_03125] SignalBasedMethodDeployment to ISignalTriggerings mapping ⌈ The SignalBasedMethodToISignalTriggeringMapping meta-class provides the ability to map a SignalBasedMethodDeployment that is referenced in the role signalBasedMethod 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::ServiceInterfaceElementToSignal Mapping			
Note	This meta-class defines the mapping of a ServiceInterface method to an ISignalTriggering. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
callSignalTriggering	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
returnSignalTriggering	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
signalBasedMethod	SignalBasedMethodDeployment	0..1	ref	Reference to the method that will be mapped to an ISignalTriggering for signal-based transport over a communication channel. **Tags:** atp.Status=draft

Table 10.9: SignalBasedMethodToISignalTriggeringMapping

Please note that the SignalBasedMethodToISignalTriggeringMapping shall also be used for the mapping of methods 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.

11 Persistency Deployment

11.1 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.

11.2 Deployment of Persistent Data

[TPS_MANI_01078] Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping ⌈ Meta-class PersistencyPortPrototypeToKeyValueDatabaseMapping has the ability to map a specific PortPrototype referenced in the role persistencyPortPrototype 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 persistencyPortPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime. ⌋*(RS_MANI_00027)*

The details can be found in Figure 11.1.

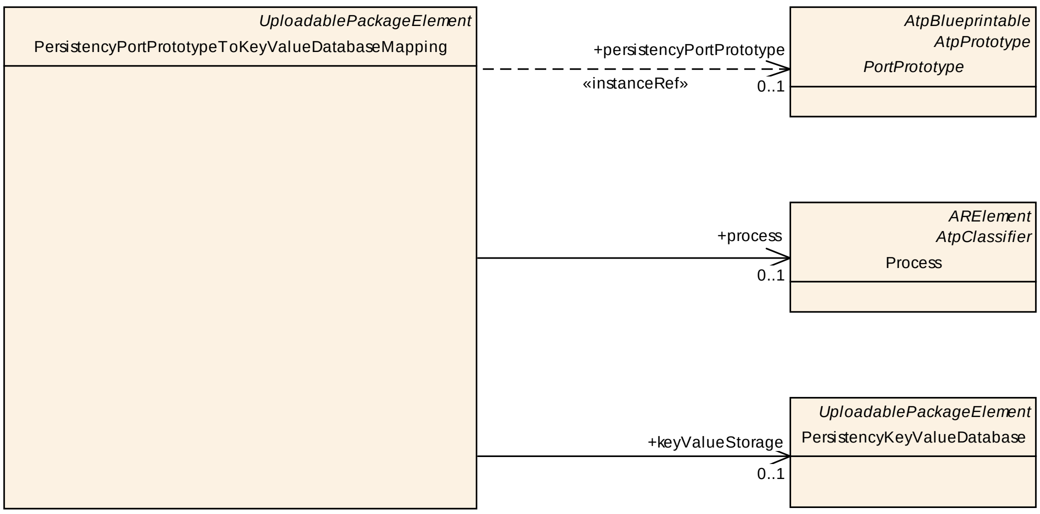

Figure 11.1: Connect a specific PortPrototype to a PersistencyKeyValueDatabase

[TPS_MANI_01079] 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::Deployment::Persistency			
Note	This meta-class represents the ability to model a key/value data base on deployment level. **Tags:** atp.Status=draft; atp.recommendedPackage=PersistencyKeyValueDatabases			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 11.1: PersistencyKeyValueDatabase

11.3 Deployment of Files

As mentioned already in chapter 3.7.3, a PortPrototype typed by a PersistencyFileProxyInterface actually builds an abstraction for an entire array of files. 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.

At one point, however, it is necessary to boil down the relation of this PortPrototype 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 PersistencyFileProxyToFileMapping, see Figure 11.2.

[TPS_MANI_01080] Semantics of PersistencyFileProxyToFileMapping ⌈
Meta-class PersistencyFileProxyToFileMapping 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)*

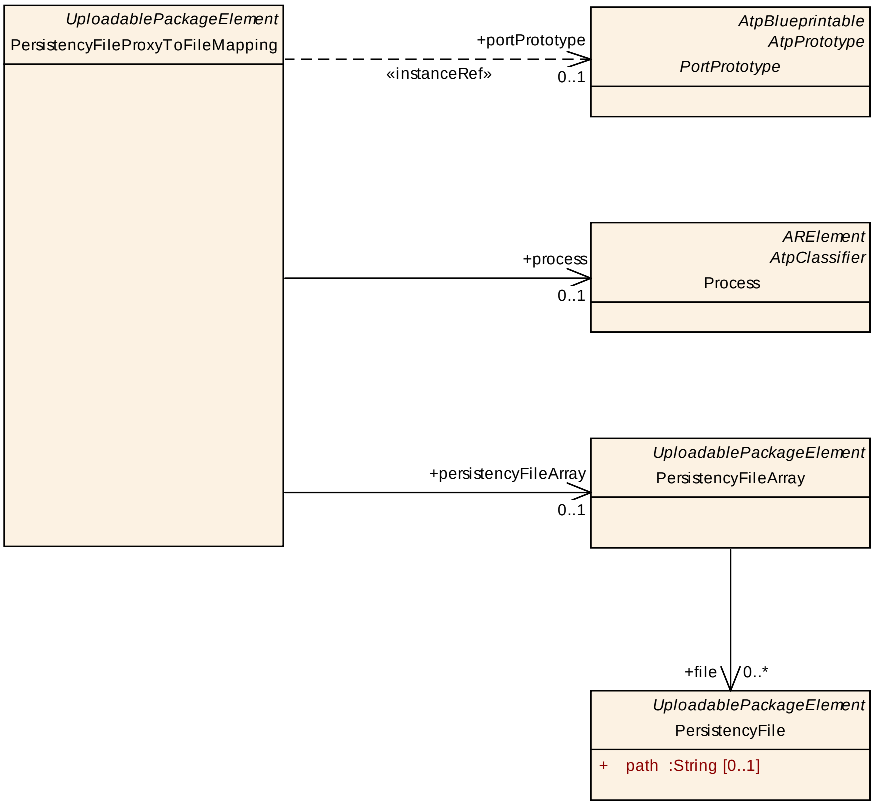

Figure 11.2: Connect a specific PortPrototype to a PersistencyFile

[constr_1525] Standardized values of PersistencyFile.category ⌈ The values of PersistencyFile.category shall be taken to further qualify the nature of the accessed files. The following values are standardized:

- TEXT_FILE
- BINARY_FILE

⌋*()*

Please note that because each PersistencyFileProxyToFileMapping.persistencyFileArray.file is eventually mapped to a specific PortPrototype typed by a PersistencyFileProxyInterface it is necessary that the value of attribute category has the same value that in turn is identical to the value of category of the referenced PortPrototype.

[constr_1526] Values of PersistencyFileArray.file.category ⌈ The value of attribute category of all PersistencyFileArray.file shall be identical and this value of attribute category in turn shall be identical to the value of the PortPrototype.category referenced by the PersistencyFileProxyToFileMapping that also references the mentioned PersistencyFileArray. ⌋*()*

The attribute PersistencyFile.path shall be taken to specify the path in the file system where the respective physical file can be found. It is obviously hard to specify this path portably relative to some other directory.

But it shall be possible to specify the path relative to the root directory "~"of the current user account.

Class	**PersistencyFile**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class represents the model of a file as part of the persistency on deployment level. **Tags:** atp.Status=draft; atp.recommendedPackage=PersistencyFiles			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
path	String	0..1	attr	This attribute holds the absolute path to the represented file on the file system. **Tags:** atp.Status=draft

Table 11.2: PersistencyFile

Class	**PersistencyFileArray**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::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.Status=draft; atp.recommendedPackage=PersistencyFileArrays			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
file	PersistencyFile	*	ref	This reference represents the collection of actual files aggregated by the PersistencyFileArray. **Tags:** atp.Status=draft

Table 11.3: PersistencyFileArray

Class	**PersistencyFileProxyToFileMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a file on deployment level to a given PortPrototype. **Tags:** atp.Status=draft; atp.recommendedPackage=PersistentFileProxyToFile Mappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
persistencyFileArray	PersistencyFileArray	0..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	0..1	ref	This reference represents the process required as context for the mapping. **Tags:** atp.Status=draft

Table 11.4: PersistencyFileProxyToFileMapping

12 Crypto Deployment

12.1 Overview

This chapter explains the part of the support for data encryption etc. in terms of mapping of concrete crypto software to the corresponding parts of the application software.

12.2 Crypto Module Instantiation

[TPS_MANI_01090] Modeling of crypto software as a platform module ⌈ An instance of the *AUTOSAR adaptive platform* hosts the crypto software as a platform module. This aspect is formalized as the definition of meta-class `CryptoModuleInstantiation` that is derived from `NonOsModuleInstantiation`.

The `CryptoModuleInstantiation`, in turn, hosts all formal elements needed to describe the deployment of crypto software and the establishment of a relation between the platform module and application-level software. ⌋*(RS_MANI_00031)*

For more information about the modeling of the `CryptoModuleInstantiation` please refer to Figure 12.1.

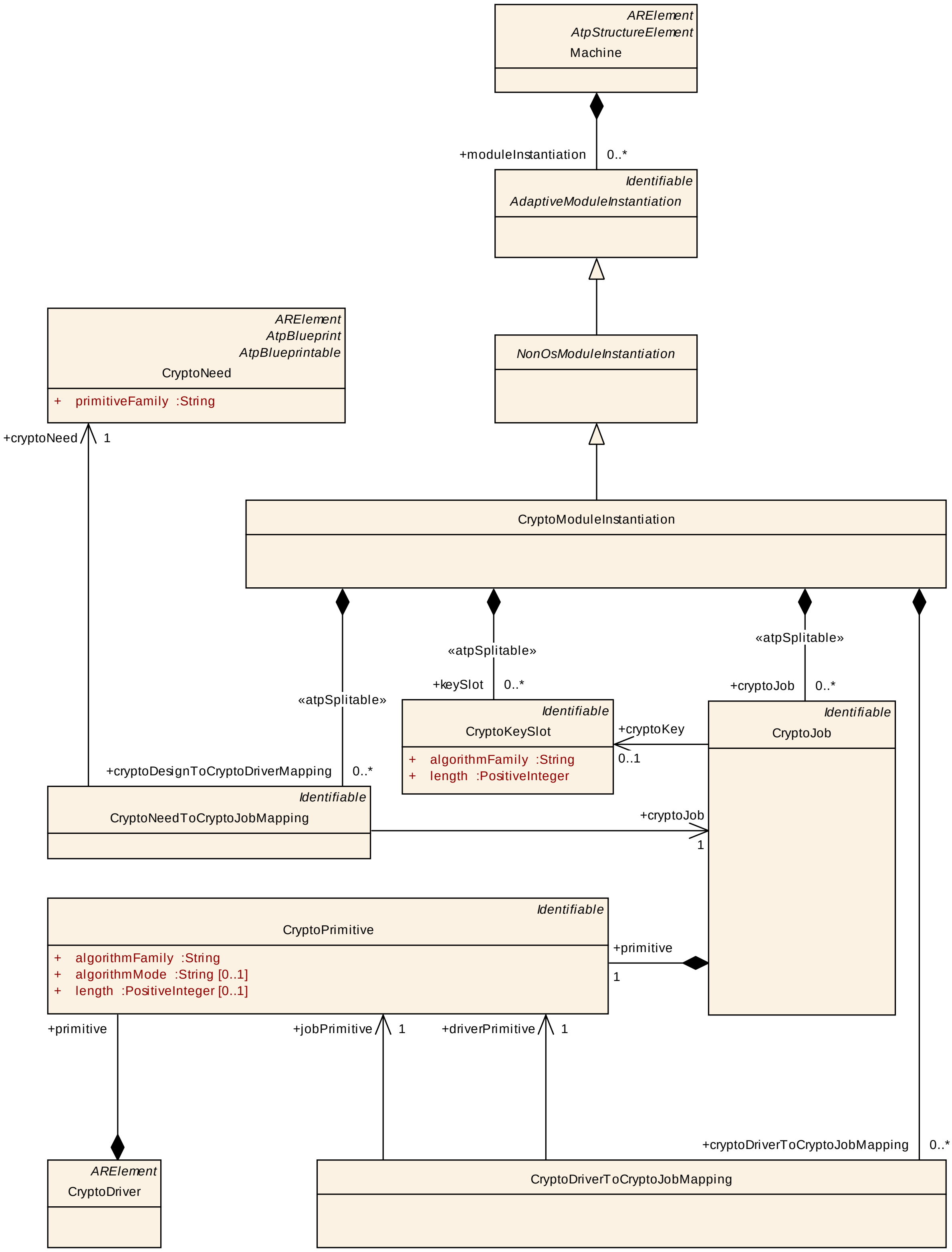

Figure 12.1: Modeling of the crypto deployment

Class	CryptoModuleInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Crypto			
Note	This meta-class represents the ability to define a concerted definition of a crypto module instantiation. **Tags:** atp.Status=draft			
Base	ARObject, AdaptiveModuleInstantiation, Identifiable, MultilanguageReferrable, NonOsModuleInstantiation, Referrable			
Attribute	Type	Mul.	Kind	Note
cryptoDesignToCryptoDriverMapping	CryptoNeedToCryptoJobMapping	*	aggr	This aggregation represents the collection of mappings from crypto job to crypto need defined in the context of the enclosing crypto module instantiation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft
cryptoDriverToCryptoJobMapping	CryptoDriverToCryptoJobMapping	*	aggr	This aggregation represents the collection of mappings from crypto primitive to crypto primitive in the context of the crypto module instantiation. **Tags:** atp.Status=draft
cryptoJob	CryptoJob	*	aggr	This aggregation represents the collection of crypto jobs defined in the context of the enclosing crypto module instantiation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft
keySlot	CryptoKeySlot	*	aggr	This aggregation represents the collection of crypto key slots defined in the context of the enclosing crypto module instantiation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft

Table 12.1: CryptoModuleInstantiation

[TPS_MANI_01095] Semantics of `CryptoKeySlot` ⌈ The actual cryptographic keys to be used by the crypto software are introduced at the production line by means of a OEM-specific workflow.

However, it is necessary to define a **representation of a cryptographic key** for the configuration of the crypto software.

This role is taken by the definition of the `CryptoKeySlot`. `CryptoModuleInstantiation` aggregates `CryptoKeySlot` in the role `keySlot`.

The properties of the `CryptoKeySlot` can be further specified by means of attributes `algorithmFamily` ([constr_1530] applies) and `length` (in bits). ⌋*(RS_MANI_00031)*

Class	CryptoKeySlot			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Crypto			
Note	This meta-class represents the ability to define a concrete key to be used for a crypto operation. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
algorithmFamily	String	1	attr	This attribute represents the ability to specify the algorithm family of the key slot. **Tags:** atp.Status=draft
length	PositiveInteger	1	attr	This attribute represents the ability to specify the length in bits of the key slot. **Tags:** atp.Status=draft

Table 12.2: CryptoKeySlot

[TPS_MANI_01096] Semantics of the `CryptoPrimitive` ⌈ The description of the cryptographic algorithm can be done by means of the `CryptoPrimitive`. The description of the cryptographic algorithm can be further supported by means of the specification of the `algorithmFamily` ([constr_1530] applies), `algorithmMode` ([constr_1531] applies), and `length`. ⌋*(RS_MANI_00031)*

[constr_1530] Standardized values of `CryptoPrimitive.algorithmFamily` and `CryptoKeySlot.algorithmFamily` ⌈ The following values of attributes `CryptoPrimitive.algorithmFamily` and `CryptoKeySlot.algorithmFamily` are standardized by AUTOSAR:

- CRYPTO_ALGOFAM_AES
- CRYPTO_ALGOFAM_3DES
- CRYPTO_ALGOFAM_PRESENT
- CRYPTO_ALGOFAM_DES
- CRYPTO_ALGOFAM_CAMELLIA
- CRYPTO_ALGOFAM_SALSA20
- CRYPTO_ALGOFAM_CHACHA20
- CRYPTO_ALGOFAM_MD5
- CRYPTO_ALGOFAM_SHA1
- CRYPTO_ALGOFAM_SHA2_256
- CRYPTO_ALGOFAM_SHA2_512
- CRYPTO_ALGOFAM_WHIRLPOOL

- CRYPTO_ALGOFAM_SHA3_256
- CRYPTO_ALGOFAM_SHA3_512
- CRYPTO_ALGOFAM_SHAKE_128
- CRYPTO_ALGOFAM_SHAKE_256
- CRYPTO_ALGOFAM_RSA
- CRYPTO_ALGOFAM_ECC

⌋*()*

[constr_1531] Standardized values of CryptoPrimitive.algorithmMode ⌈ The following values of attribute CryptoPrimitive.algorithmMode are standardized by AUTOSAR:

- CRYPTO_ALGOMODE_ECB
- CRYPTO_ALGOMODE_CRC
- CRYPTO_ALGOMODE_CTR
- CRYPTO_ALGOMODE_GCM
- CRYPTO_ALGOMODE_CCM
- CRYPTO_ALGOMODE_STREAM
- CRYPTO_ALGOMODE_STREAM_POLY1305
- CRYPTO_ALGOMODE_HMAC
- CRYPTO_ALGOMODE_CMAC
- CRYPTO_ALGOMODE_PLOY1305
- CRYPTO_ALGOMODE_MIYAGUCHI_PRENEEL
- CRYPTO_ALGOMODE_HIROSE
- CRYPTO_ALGOMODE_PKCS_V15
- CRYPTO_ALGOMODE_PKCS_V2
- CRYPTO_ALGOMODE_ECDSA
- CRYPTO_ALGOMODE_EDDSA
- CRYPTO_ALGOMODE_ECIES_X963

⌋*()*

Class	CryptoPrimitive			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Crypto			
Note	This meta-class represents the ability to describe a crypto algorithm in an abstract form. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
algorithmFamily	String	1	attr	This attribute represents the ability to specify the algorithm family of the crypto primitive. **Tags:** atp.Status=draft
algorithmMode	String	0..1	attr	This attribute represents the ability to specify the algorithm mode of the crypto primitive. **Tags:** atp.Status=draft
length	PositiveInteger	0..1	attr	This attribute represents the ability to specify the length in bits on which the crypto primitive is operating. **Tags:** atp.Status=draft

Table 12.3: CryptoPrimitive

Beyond the regulation made by [constr_1530] and [constr_1531] it is possible to assign custom values to `CryptoPrimitive.algorithmFamily` and `CryptoKeySlot.algorithmFamily` resp. `CryptoPrimitive.algorithmMode`.

In this case, however, it is mandatory to use a company-specific prefix or suffix to the custom values in order to positively avoid clashes with potential future extensions of the collection of standardized values defined by [constr_1530] and [constr_1531].

12.3 Crypto Job

[TPS_MANI_01091] Semantics of `CryptoJob` ⌈ The formal definition of a `CryptoJob` represents a specific usage of (or call to) a cryptographic software function. This software function is part of the `CryptoModuleInstantiation`, hence the aggregation of `CryptoJob` at `CryptoModuleInstantiation` in the role `cryptoJob`. ⌋ *(RS_MANI_00031)*

A `CryptoJob` is defined by the implemented crypto algorithm (modeled as a `CryptoPrimitive`) as well as the used `CryptoKeySlot`. This relation (as depicted in Figure 12.1) is modeled by the aggregation of the `primitive` as well as the reference to the `cryptoKey`.

Class	**CryptoJob**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Crypto			
Note	This meta-class represents the ability to model a crypto job. The latter in turn 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.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
cryptoKey	CryptoKeySlot	0..1	ref	This represents the key slots to which the referencing crypto job applies. **Tags:** atp.Status=draft
primitive	CryptoPrimitive	1	aggr	This aggregation defines the crypto primitive applicable for the enclosing crypto job. **Tags:** atp.Status=draft

Table 12.4: CryptoJob

[TPS_MANI_01092] Mapping between `CryptoNeed` and `CryptoJob` ⌈ It is necessary to create a formal relation between a `CryptoNeed` formulated by an OEM to a `CryptoJob` (which is typically defined in the domain of a supplier). The formalization of this relation is the `CryptoNeedToCryptoJobMapping`.

By means of this mapping in combination with the `CryptoNeedToPortPrototypeMapping` it is possible to define the relation of a `PortPrototype` in the application software to the corresponding `CryptoJob` in the platform software.

In other words, the `ClientServerOperation`s called by the application software are (typically by means of an IPC mechanism) redirected to the corresponding `CryptoJob`. ⌋*(RS_MANI_00031)*

Class	**CryptoNeedToCryptoJobMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Crypto			
Note	This meta-class represents the ability to define a mapping from crypto need to crypto job. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
cryptoJob	CryptoJob	1	ref	This represents the crypto job part of the mapping from crypto need to crypto job. **Tags:** atp.Status=draft
cryptoNeed	CryptoNeed	1	ref	This represents the crypto need part of the mapping from crypto need to crypto job. **Tags:** atp.Status=draft

Table 12.5: CryptoNeedToCryptoJobMapping

12.4 Crypto Driver

[TPS_MANI_01093] Semantics of CryptoDriver ⌈ The CryptoDriver represents an abstraction around details of the implementation of crypto routines.

For example, the existence of a CryptoDriver is supposed to make the upper layer software independent of the question whether the crypto functionality is implemented by means of pure software or whether some parts are taken over by a hardware component. ⌋*(RS_MANI_00031)*

Of course, the CryptoDriver has a strong relation to the underlying crypto algorithm, thus the aggregation the CryptoPrimitive in the role primitive.

Class	**CryptoDriver**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Crypto			
Note	This meta-class represents the ability to model a crypto driver. **Tags:** atp.Status=draft; atp.recommendedPackage=CryptoDrivers			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
primitive	CryptoPrimitive	1	aggr	This aggregation represents the collection of crypto primitives in the context of the enclosing crypto driver. **Tags:** atp.Status=draft

Table 12.6: CryptoDriver

[TPS_MANI_01094] Scope of CryptoDriver ⌈ The CryptoDriver is derived from ARElement. It is not part of the CryptoModuleInstantiation because the same CryptoDriver could be used on different Machines (at the same time).

Consequently, the actual relation between a given CryptoDriver and a specific Machine is **indirectly** created by means of the CryptoDriverToCryptoJobMapping. ⌋*(RS_MANI_00031)*

For clarification, the CryptoDriverToCryptoJobMapping references a CryptoPrimitive in the role driverPrimitive and therefore the specific CryptoDriver that aggregates the referenced driverPrimitive is also unambiguously identified.

Obviously, the same argumentation applies for the reference CryptoDriverToCryptoJobMapping.jobPrimitive. The referenced CryptoJob is a member of an aggregation chain the finally ends at the Machine. Therefore, by referencing the CryptoJob the applicable Machine is unambiguously identified.

The actual motivation for the existence of this “indirect” mapping goes down to the fact that the CryptoDriverToCryptoJobMapping (that references two CryptoPrimitives) very much facilitates the check for consistency in terms of whether the referenced CryptoPrimitives fit to each other.

Class	**CryptoDriverToCryptoJobMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Crypto			
Note	This meta-class has the ability to map two crypto primitives onto each other. This mapping effectively also maps a crypto driver to a crypto job. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
driverPrimitive	CryptoPrimitive	1	ref	This reference represents the crypto driver in the context of the mapping of two crypto primitives. **Tags:** atp.Status=draft
jobPrimitive	CryptoPrimitive	1	ref	This reference represents the crypto job in the context of the mapping of two crypto primitives. **Tags:** atp.Status=draft

Table 12.7: CryptoDriverToCryptoJobMapping

13 Secure Communication Deployment

13.1 Overview

This chapter explains the part of using concrete crypto software to realize secured communication etc. in terms of mapping of SecureComProps to concrete CryptoJobs and CryptoKeySlots.

For each supported secure communication protocol an own SecureCommunicationDeployment specialization exists that will be explained in the following subchapters.

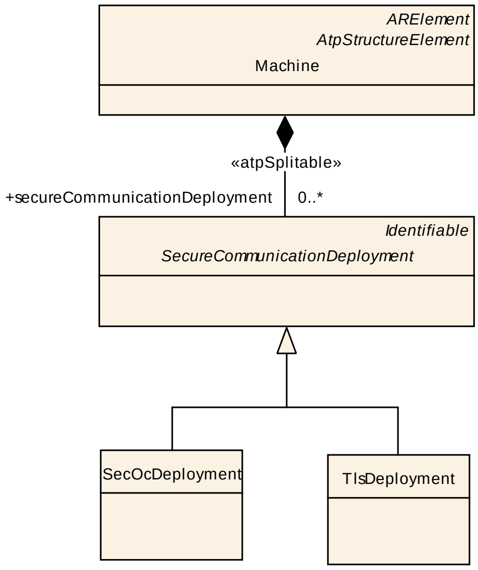

Figure 13.1: Modeling of the secure communication deployment

Class	**SecureCommunicationDeployment (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	The meta-class represents the ability to define a deployment of secure communication protocol configuration settings to crypto module entities. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 13.1: SecureCommunicationDeployment

13.2 SecOc Deployment

The SecOcDeployment describes the realization of SecOC secured communication by using a crypto platform module that is described as CryptoModuleInstantiation as defined in [TPS_MANI_01090]. In case of SecOC the crypto platform module provides cryptographic algorithms to generate and verify Cryptographic Signatures or Message Authentication Codes.

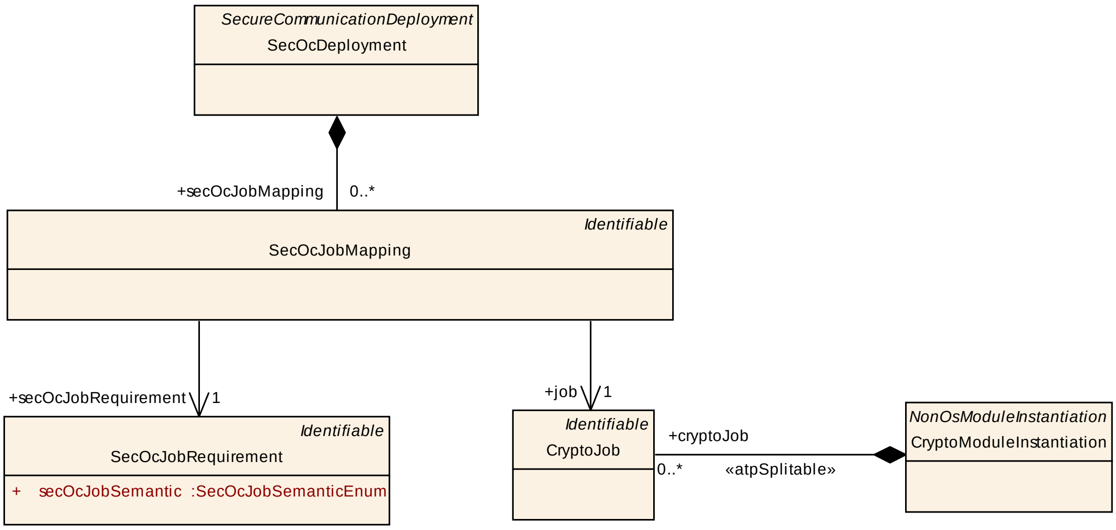

Figure 13.2: Modeling of the SecOC secure communication deployment

Class	**SecOcDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::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.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, SecureCommunication Deployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
secOcJob Mapping	SecOcJobMapp ing	*	aggr	Mapping of the JobRequirement to a concrete crypto job. **Tags:** atp.Status=draft

Table 13.2: SecOcDeployment

[TPS_MANI_03141] Mapping between `SecOcJobRequirement` and `CryptoJob` ⌈
It is necessary to create a formal relation between a `SecOcJobRequirement` that is formulated in most cases by an OEM to a `CryptoJob` (which is typically defined in the domain of a supplier). The formalization of this relation is the `SecOcJobMapping`. ⌋ *(RS_MANI_00036, RS_MANI_00031)*

Class	**SecOcJobMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	This meta-class allows to map a SecOcJobRequirement to a concrete crypto job that will fulfill the JobRequirement. 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.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
job	CryptoJob	1	ref	Reference to a concrete CryptoJob defined in the CryptoDeployment. **Tags:** atp.Status=draft
secOcJobRequirement	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 13.3: SecOcJobMapping

13.3 TLS Deployment

The `TlsDeployment` describes the realization of Tls secured communication by using a crypto platform module that is described as `CryptoModuleInstantiation` as defined in [TPS_MANI_01090]. In case of Tls the crypto platform module provides cryptographic algorithms to generate and verify Cryptographic Signatures or Message Authentication Codes and to encrypt and decrypt the data. In addition a key management is provided.

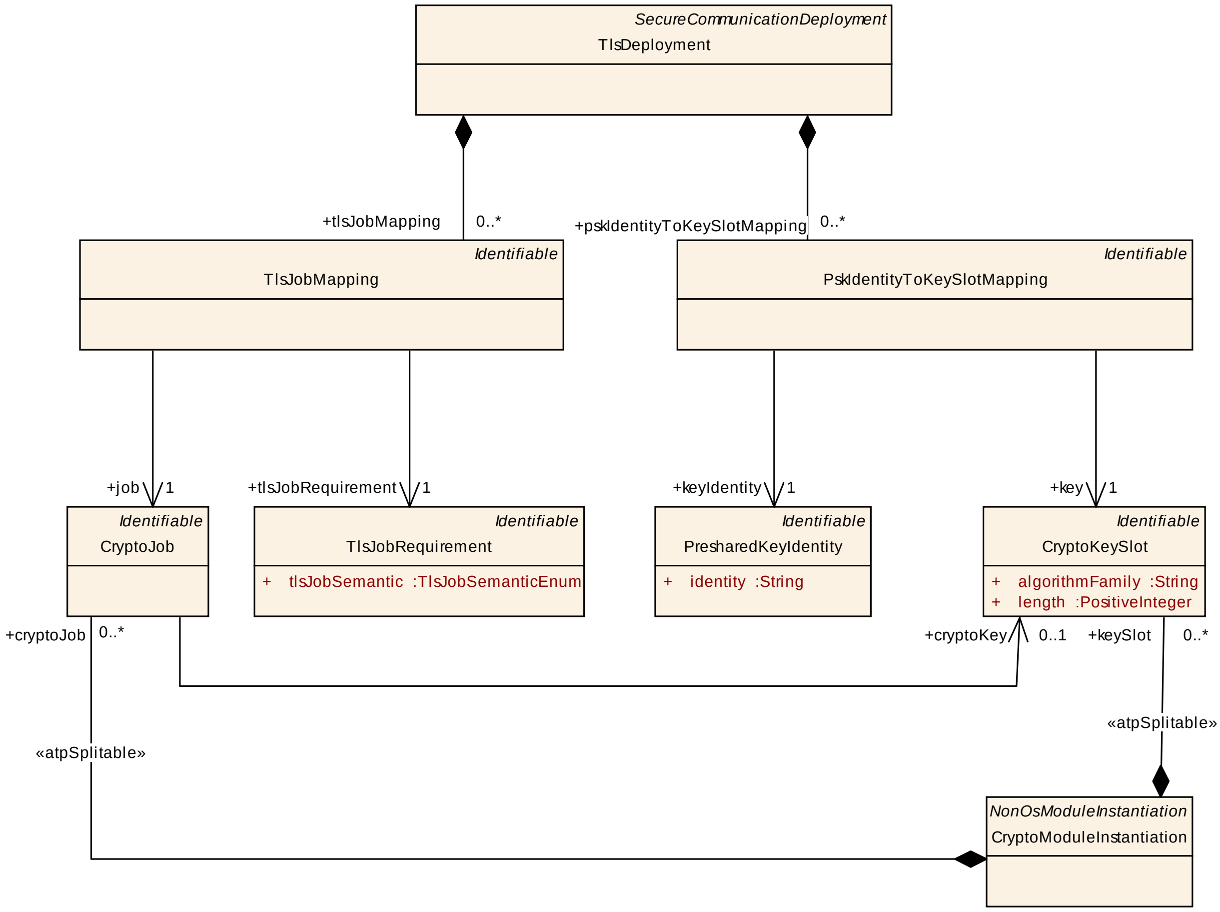

Figure 13.3: Modeling of the TLS secure communication deployment

Class	**TlsDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::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.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 13.4: TlsDeployment

[TPS_MANI_03142] Mapping between `TlsJobRequirement` and `CryptoJob` ⌈ It is necessary to create a formal relation between a `TlsJobRequirement` that is formulated in most cases by an OEM to a CryptoJob (which is typically defined in the domain of a supplier). The formalization of this relation is the TlsJobMapping. ⌋ (*RS_MANI_00031, RS_MANI_00036*)

Class	TlsJobMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	This meta-class allows to map a TlsJobRequirement to a concrete crypto job that will fulfill the JobRequirement. 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.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
job	CryptoJob	1	ref	Reference to a concrete CryptoJob defined in the CryptoDeployment. **Tags:** atp.Status=draft
tlsJobRequirement	TlsJobRequirement	1	ref	Reference to a TLS JobRequirement that defines requirements for the cryptographic job that need to be executed. **Tags:** atp.Status=draft

Table 13.5: TlsJobMapping

The CryptoKeySlot defines a representation of a cryptographic key in the configuration of the crypto software as defined in [TPS_MANI_01095]. TLS pre-shared keys are shared between the communicating parties to establish a TLS connection. To find the key that corresponds to the PSK identity the PskIdentityToKeySlotMapping is introduced.

[TPS_MANI_03143] Mapping between PresharedKeyIdentity and CryptoKeySlot ⌈ Meta-class PskIdentityToKeySlotMapping has the ability to map a PresharedKeyIdentity defined in TlsSecureComProps to a CryptoKeySlot defined in CryptoModuleInstantiation. ⌋(*RS_MANI_00031, RS_MANI_00036*)

Class	PskIdentityToKeySlotMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::SecureCommunication			
Note	This meta-class allows to map a PresharedKeyIdentity to a concrete key that will be used for a crypto operation. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
key	CryptoKeySlot	1	ref	Reference to a concrete key defined in the CryptoDeployment. **Tags:** atp.Status=draft

keyIdentity	PresharedKeyIdentity	1	ref	Reference to a Preshared Key Identity. **Tags:** atp.Status=draft

Table 13.6: PskIdentityToKeySlotMapping

14 Platform Health Management Deployment

14.1 Overview

This chapter explains the interaction of application software with the platform health management deployment.

An application software can define the usage of several platform health management supervision entities and checkpoints (see chapter 3.8.5.1). In order to define the interaction between the application software and the platform health management the connections between their PortPrototypes have to be described during deployment(see figure 14.1).

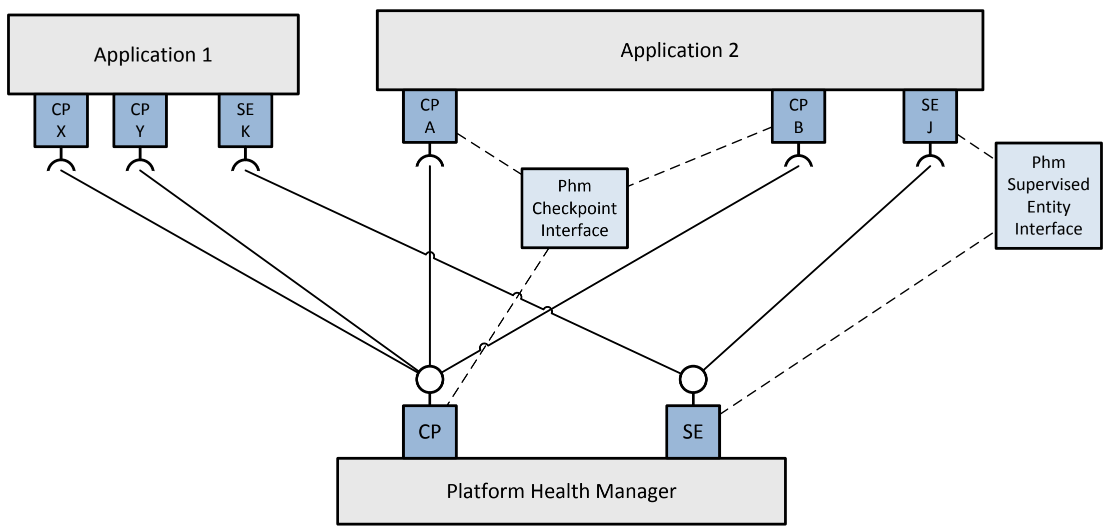

Figure 14.1: Interaction of application software with the platform health manager

The PlatformHealthManagementContribution allows to describe aspects for the deployment of requirements how the platform health management shall behave during runtime.

[TPS_MANI_03502] 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)*

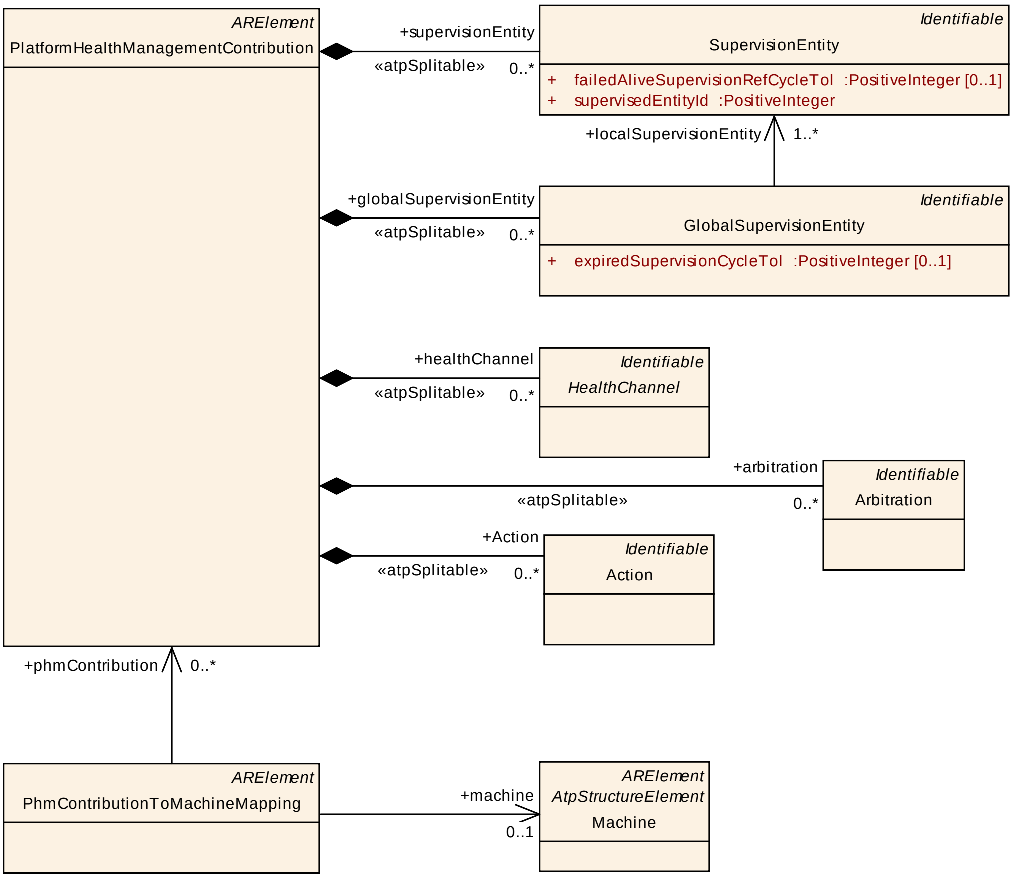

Figure 14.2: Modeling of PlatformHealthManagementContribution

The PlatformHealthManagementContribution is structured into several aspects which will be described in the following sections:

- Supervision entity (section 14.2)
- Health channels (section)
- Arbitration and rules (section)
- Actions (section)

Class	**PlatformHealthManagementContribution**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines a contribution to the Platform Health Management. **Tags:** atp.Status=draft; atp.recommendedPackage=PlatformHealthManagement Contributions			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

Action	Action	*	aggr	Collection of Actions and ActionLists in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft xml.sequenceOffset=50
arbitration	Arbitration	*	aggr	Collection of Arbitrations in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft xml.sequenceOffset=40
globalSupervisionEntity	GlobalSupervisionEntity	*	aggr	Collection of GlobalSupervisionEntitys in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft xml.sequenceOffset=20
healthChannel	HealthChannel	*	aggr	Collection of HealthChannels in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft xml.sequenceOffset=30
supervisionEntity	SupervisionEntity	*	aggr	Collection of SupervisionEntitys in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft xml.sequenceOffset=10

Table 14.1: PlatformHealthManagementContribution

Class	**PhmContributionToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement			
Note	This element associates one or more PlatformHealthManagementContributions with a Machine. **Tags:** atp.Status=draft; atp.recommendedPackage=PhmContributionToMachineMappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, 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

phmContribution	PlatformHealthManagementContribution	*	ref	This reference identifies one or more PlatformHealthManagementContributions in the context of a PhmContributionToMachineMapping. **Tags:** atp.Status=draft

Table 14.2: PhmContributionToMachineMapping

In chapter 3.8.5.1 it is explained that the application software implementation just calls methods on the respective `PortPrototype`s to interact with the platform health management. From the application developer these methods have no addressing information, because the identity of the `PortPrototype` is the identification in the scope of the application software.

The deployed structure (according to figure 14.2) however requires more information when an `API` at the platform health manager is called, namely:

- `SupervisionEntity.supervisedEntityId`
- `PhmCheckpoint.checkpointId`
- 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 connections from figure 14.2). The order of this argument injection is determined by the specification of the platform health management.

14.2 Supervision entity deployment

In the application design chapter of this document the declaration of supervised entities and checkpoints has been described (see section 3.8.5.1). These declarations provide the view on supervision from the application software code point.

For the configuration of the platform health management the definition of `SupervisionEntity` and `PhmCheckpoint` are used to stand in for the corresponding supervised entities and checkpoints of the application design.

[TPS_MANI_03504] Existence of `SupervisionEntity` ⌈ For each supervised entity in the application definition there may be a `SupervisionEntity` defined. The correspondence of the two is defined by the instance reference `SupervisionEntity.supervisedEntity` ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03505] Existence of `PhmCheckpoint` ⌈ For each checkpoint in the application definition there may be a `PhmCheckpoint` defined. The correspondence of the two is defined by the instance reference `PhmCheckpoint.checkpoint` ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03506] Optionality of SupervisionEntity and PhmCheckpoint ⌈ It is not required that every supervised entity or checkpoint of the application definition eventually has a corresponding `SupervisionEntity` or `PhmCheckpoint` 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)*

Class	**SupervisionEntity**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines a supervision entity in the context of platform health management contribution. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
aliveSupervision	AliveSupervision	*	aggr	Collection of AliveSupervisions in the context of this SupervisionEntity. **Tags:** atp.Status=draft
checkpoint	PhmCheckpoint	*	aggr	Collection of PhmCheckpoints in the context of this SupervisionEntity. **Tags:** atp.Status=draft
deadlineSupervision	DeadlineSupervision	*	aggr	Collection of DeadlineSupervisions in the context of this SupervisionEntity. **Tags:** atp.Status=draft
failedAliveSupervisionRefCycleTol	PositiveInteger	0..1	attr	Defines the acceptable amount of cycles with incorrect/failed alive supervisions for this SupervisionEntity before it is considered failed. **Tags:** atp.Status=draft
logicalSupervision	LogicalSupervision	*	aggr	Collection of LogicalSupervisions in the context of this SupervisionEntity. **Tags:** atp.Status=draft
process	Process	0..1	ref	Reference to the process this SupervisionEntity shall be applied to. **Tags:** atp.Status=draft
supervisedEntity	RPortPrototype	0..1	iref	Reference to a RPortPrototype representing a supervised entity for Platform Health Management. **Tags:** atp.Status=draft
supervisedEntityId	PositiveInteger	1	attr	Defines the Id to be used by the calling application when the respective SupervisedEntity is referred to. **Tags:** atp.Status=draft

transition	CheckpointTransition	*	aggr	Collection of CheckpointTransitions in the context of this SupervisionEntity. **Tags:** atp.Status=draft

Table 14.3: SupervisionEntity

Class	**PhmCheckpoint**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement			
Note	This element contains an instance reference to a RPortPrototype representing a checkpoint for Platform Health Management. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
checkpoint	RPortPrototype	1	iref	Reference to a RPortPrototype representing a checkpoint for Platform Health Management. **Tags:** atp.Status=draft
checkpointId	PositiveInteger	1	attr	Defines the Id to be used by the calling application when this checkpoint is referred to. **Tags:** atp.Status=draft

Table 14.4: PhmCheckpoint

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 (`ModeDependentStartupConfig`) and with which arguments (`StartupOption`) the `Executable` will be started.

Based on the conditions and arguments the behavior of the application may change and thus also the supervision of the application may need to be different.

[TPS_MANI_03503] Applicability of supervision to a specific `Process` ⌈The reference `SupervisionEntity.process` defines to which specific `Process` this `SupervisionEntity` definition shall be applied to. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03515] Expiration tolerance for `SupervisionEntity` ⌈The attribute `SupervisionEntity.failedAliveSupervisionRefCycleTol` 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)*

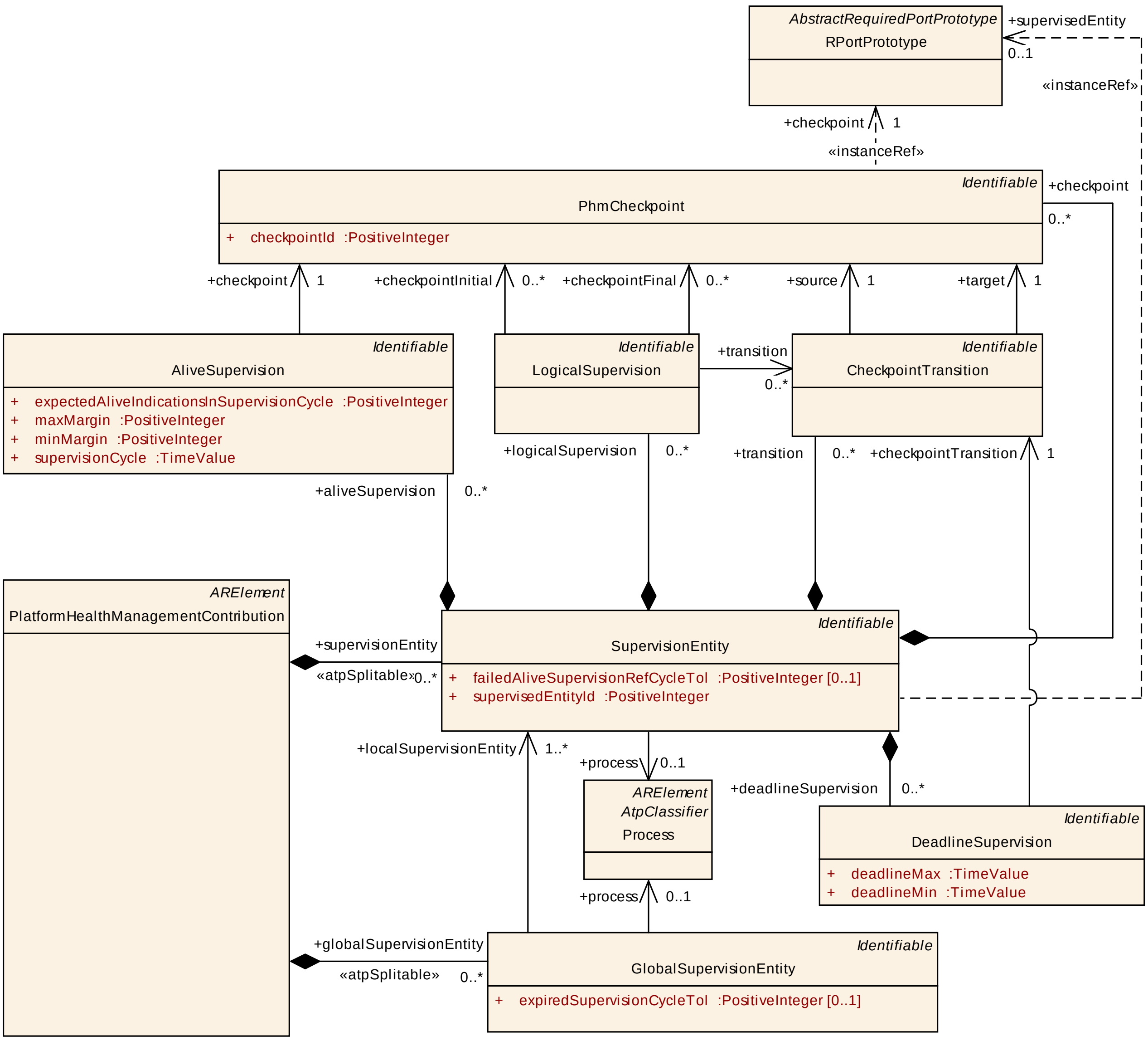

Figure 14.3: Modeling of SupervisionEntity

14.2.1 AliveSupervision definition

In the scope of a SupervisionEntity an AliveSupervision can be defined for a specific PhmCheckpoint. SupervisionEntity can be used to define in which timing boundaries one specific checkpoint shall be monitored.

[TPS_MANI_03508] Definition of an AliveSupervision for a PhmCheckpoint ⌈
An AliveSupervision definition provides attributes to configure the supervision of the referenced PhmCheckpoint.

- supervisionCycle defines the time base used monitor the reporting of this specific PhmCheckpoint

- expectedAliveIndicationsInSupervisionCycle defines the number of indications which shall be observed during the time period defined by supervisionCycle
- minMargin and maxMargin define the acceptable deviation from the expectedAliveIndicationsInSupervisionCycle within the time period defined by supervisionCycle

⌋(RS_MANI_00023, RS_MANI_00032)

Class	**AliveSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	Defines an AliveSupervision for one checkpoint. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
checkpoint	PhmCheckpoint	1	ref	Reference to a checkpoint in the context of AliveSupervision. **Tags:** atp.Status=draft
expectedAliveIndicationsInSupervisionCycle	PositiveInteger	1	attr	Defines the amount of expected alive indications of the checkpoint within the supervisionCycle. **Tags:** atp.Status=draft
maxMargin	PositiveInteger	1	attr	Defines the amount of alive indications of the checkpoint that are acceptable to be more than the expected alive indications within the supervisionCycle. **Tags:** atp.Status=draft
minMargin	PositiveInteger	1	attr	Defines the amount of alive indications of the checkpoint that are acceptable to be less than the expected alive indications within the supervisionCycle. **Tags:** atp.Status=draft
supervisionCycle	TimeValue	1	attr	Defines the time base for the supervision cycle. **Tags:** atp.Status=draft

Table 14.5: AliveSupervision

14.2.2 CheckpointTransition definition

For the definition of further supervision strategies the need to first define possible CheckpointTransitions between PhmCheckpoints 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 SupervisionEntity and can be used by the LogicalSupervision and DeadlineSupervision.

[TPS_MANI_03509] Definition of a CheckpointTransition ⌈ A CheckpointTransition defines one possible transition from the source PhmCheckpoint to the target PhmCheckpoint. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**CheckpointTransition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	Defines one transition between two checkpoints. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
source	PhmCheckpoint	1	ref	Reference to the source checkpoint for this transition. **Tags:** atp.Status=draft
target	PhmCheckpoint	1	ref	Reference to the target checkpoint for this transition. **Tags:** atp.Status=draft

Table 14.6: CheckpointTransition

14.2.3 LogicalSupervision definition

The LogicalSupervision defines a 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.

When a PhmCheckpoint is reported to the platform health management where there is no CheckpointTransition defined from the last reported PhmCheckpoint as source to the current reported PhmCheckpoint as target, this situation violates the LogicalSupervision.

[TPS_MANI_03510] Definition of LogicalSupervision ⌈ A LogicalSupervision defines relations between PhmCheckpoints which form a directed graph from one or more checkpointInitial PhmCheckpoints through a set of CheckpointTransitions defined by collection of transitions to one or more checkpointFinal PhmCheckpoints. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**LogicalSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	Defines a LogicalSupervision graph consisting of transitions, initial- and final checkpoints. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
checkpoint Final	PhmCheckpoint	*	ref	Reference to the final Checkpoint(s) for this LogicalSupervision. **Tags:** atp.Status=draft xml.sequenceOffset=20
checkpoint Initial	PhmCheckpoint	*	ref	Reference to the initial Checkpoint(s) for this LogicalSupervision. **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 14.7: LogicalSupervision

14.2.4 `DeadlineSupervision` definition

The `DeadlineSupervision` defines timing attributes for one specific `CheckpointTransition`.

[TPS_MANI_03511] 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::Deployment::PlatformHealth Management			
Note	Defines an DeadlineSupervision for one transition. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
checkpoint Transition	CheckpointTransition	1	ref	Reference to the transition in the context of a DeadlineSupervision. **Tags:** atp.Status=draft

deadlineMax	TimeValue	1	attr	Defines the longest time span before which the deadline is considered to be met for transition. **Tags:** atp.Status=draft
deadlineMin	TimeValue	1	attr	Defines the shortest time span after which the deadline is considered to be met for transition. **Tags:** atp.Status=draft

Table 14.8: DeadlineSupervision

14.3 Global supervision entity deployment

The `GlobalSupervisionEntity` definition of supervision for the platform health management is a second level supervision which takes the result of one or several `SupervisionEntity`s (with their respective `AliveSupervision`s, `LogicalSupervision`s, and `DeadlineSupervision`s) and aggregates the individual statuses of these supervisions into one global supervision status (see also figure 14.3).

[TPS_MANI_03513] Collection of `SupervisionEntity`s into a global supervision ⌈ All referenced `SupervisionEntity`s in the scope of `GlobalSupervisionEntity.localSupervisionEntity` shall be taken into the aggregation of the status of the `GlobalSupervisionEntity`. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03512] Applicability of global supervision to a specific `Process` ⌈ The reference `GlobalSupervisionEntity.process` defines to which specific `Process` this `GlobalSupervisionEntity` definition shall be applied to. ⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03514] Expiration tolerance for `GlobalSupervisionEntity` ⌈ The attribute `GlobalSupervisionEntity.expiredSupervisionCycleTol` defines how many supervision cycles this incorrect global supervision is maintained in the state *failed* before it is considered *expired*. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**GlobalSupervisionEntity**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement			
Note	This element defines a collection of SupervisionEntitys in order to provide a aggregated supervision state of the referenced SupervisionEntitys. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
expiredSupervisionCycleTol	PositiveInteger	0..1	attr	Defines the acceptable amount of cycles with incorrect/failed alive supervisions for this GlobalSupervisionEntity before it is considered expired. **Tags:** atp.Status=draft

localSupervisionEntity	SupervisionEntity	1..*	ref	Reference to the SupervisionEntitys which shall be considered for this GlobalSupervisionEntity. **Tags:** atp.Status=draft
process	Process	0..1	ref	Reference to the process this GlobalSupervisionEntity shall be applied to. **Tags:** atp.Status=draft

Table 14.9: GlobalSupervisionEntity

14.4 Health channel deployment

The `HealthChannel` is used as an abstraction to the platform health management input for the arbitration and rule evaluation (see chapter 14.5).

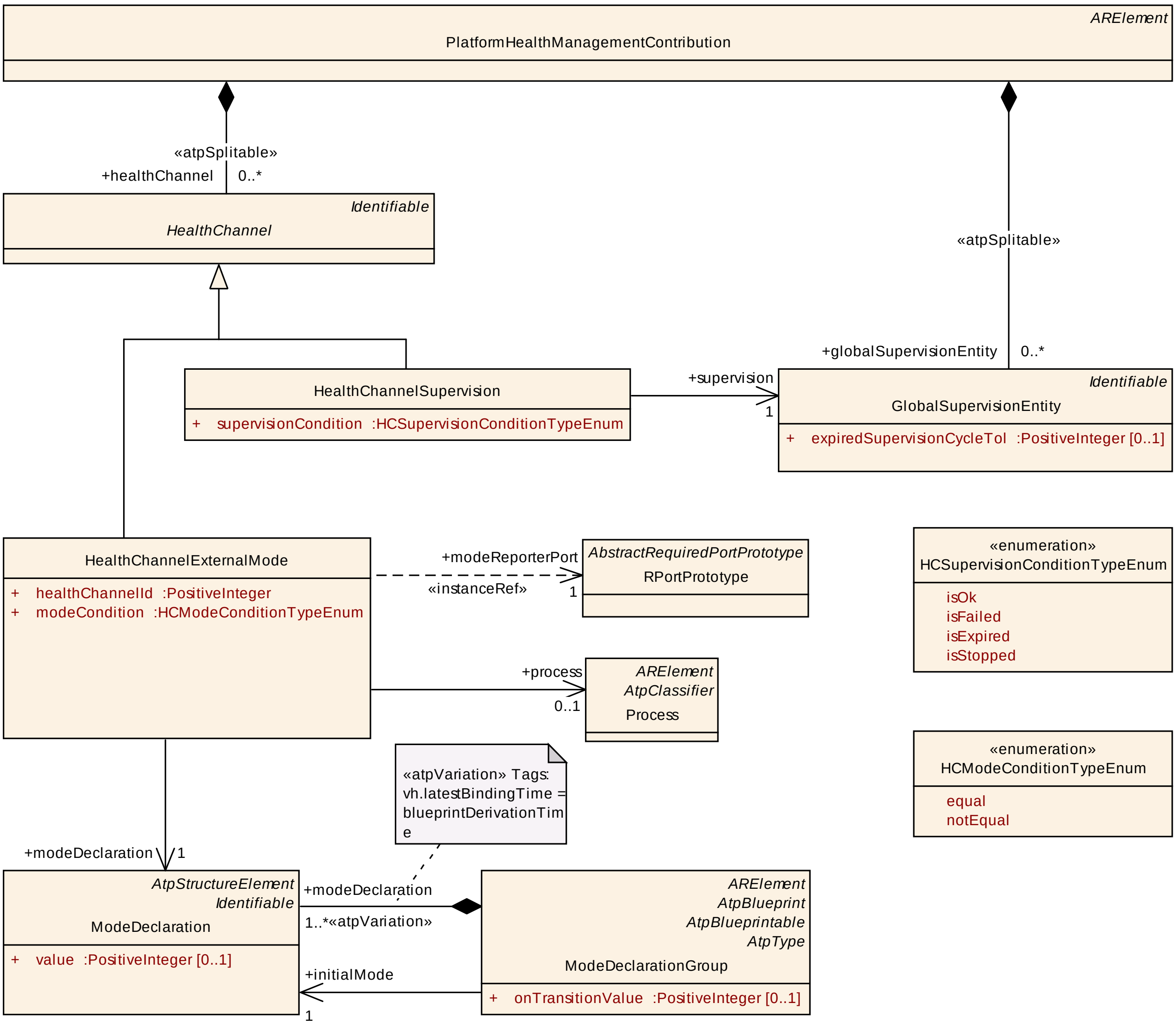

Figure 14.4: Modeling of `HealthChannel`

Class	HealthChannel (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines the source of a health channel. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 14.10: HealthChannel

The specialized use-cases for HealthChannels are describe in the following sections.

14.4.1 Supervision health channel deployment

The HealthChannelSupervision HealthChannel is used to compare the status of a GlobalSupervisionEntity with a constant status and provide the result as input to the platform health management arbitration engine.

[TPS_MANI_03516] Condition evaluation for HealthChannelSupervision ⌈The status of the GlobalSupervisionEntity 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::Deployment::PlatformHealth Management			
Note	This element defines a health channel representing the status of a GlobalSupervisionEntity. **Tags:** atp.Status=draft			
Base	ARObject, HealthChannel, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
supervision	GlobalSupervisionEntity	1	ref	Reference to the GlobalSupervisionEntity as source for the health channel. **Tags:** atp.Status=draft
supervisionCondition	HCSupervisionConditionTypeEnum	1	attr	Defines which condition shall trigger this health channel wrt. the referenced GlobalSupervisionEnity. **Tags:** atp.Status=draft

Table 14.11: HealthChannelSupervision

Enumeration	**HCSupervisionConditionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management
Note	Defines the possible conditions which can be evaluated in the scope of a GlobalSupervisionEntity. **Tags:** atp.Status=draft
Literal	***Description***
isExpired	**Tags:** atp.EnumerationValue=2
isFailed	**Tags:** atp.EnumerationValue=1
isOk	**Tags:** atp.EnumerationValue=0
isStopped	**Tags:** atp.EnumerationValue=3

Table 14.12: HCSupervisionConditionTypeEnum

14.4.2 External mode health channel deployment

The HealthChannelExternalMode HealthChannel is used to compare a reported mode to a constant mode declaration and provide the result as input to the platform health management arbitration engine.

[TPS_MANI_03517] Condition evaluation for HealthChannelExternalMode ⌈ The reported value of the HealthChannelExternalMode which is referenced in the role modeReporterPort will be compared to the constant status provided in modeDeclaration. The modeCondition defines whether it shall be compared for equality or non-equality. The result of this comparison is then the result of the HealthChannelExternalMode evaluation. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**HealthChannelExternalMode**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines a health channel representing the status of a mode. **Tags:** atp.Status=draft			
Base	ARObject, HealthChannel, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
healthChannelId	PositiveInteger	1	attr	Defines the Id to be used by the calling application when this HealthChannel is referred to. **Tags:** atp.Status=draft
modeCondition	HCModeConditionTypeEnum	1	attr	Defines which condition shall trigger this health channel wrt. the referenced mode. **Tags:** atp.Status=draft

modeDeclaration	ModeDeclaration	1	ref	Reference to the ModeDeclaration which is taken as the comparator for the health channel's reported mode. **Tags:** atp.Status=draft
modeReporterPort	RPortPrototype	1	iref	Reference to a port where the mode will be reported for this health channel. **Tags:** atp.Status=draft
process	Process	0..1	ref	Reference to a process in the scope of a HealthChannelExternalMode. **Tags:** atp.Status=draft

Table 14.13: HealthChannelExternalMode

Enumeration	**HCModeConditionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management
Note	Defines the possible conditions which can be evaluated in the scope of a mode. **Tags:** atp.Status=draft
Literal	***Description***
equal	**Tags:** atp.EnumerationValue=0
notEqual	**Tags:** atp.EnumerationValue=1

Table 14.14: HCModeConditionTypeEnum

In chapter 3.8.5.1 it is explained that the application software implementation just calls methods on the respective `PortPrototype`s to interact with the platform health management. From the application developer these methods have no addressing information, because the identity of the `PortPrototype` is the identification in the scope of the application software.

The deployed structure (according to figure 14.2) however requires more information when an `API` at the platform health manager is called, namely:

- `HealthChannelExternalMode.healthChannelId`
- 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 connections from figure 14.2). The order of this argument injection is determined by the specification of the platform health management.

14.5 Arbitration and rule deployment

The Arbitration defines the expressions and rules to calculate a logical statement from a set of input HealthChannels. The results of these calculations are used to define the triggering of specific actions by the platform health management (see chapter 14.6).

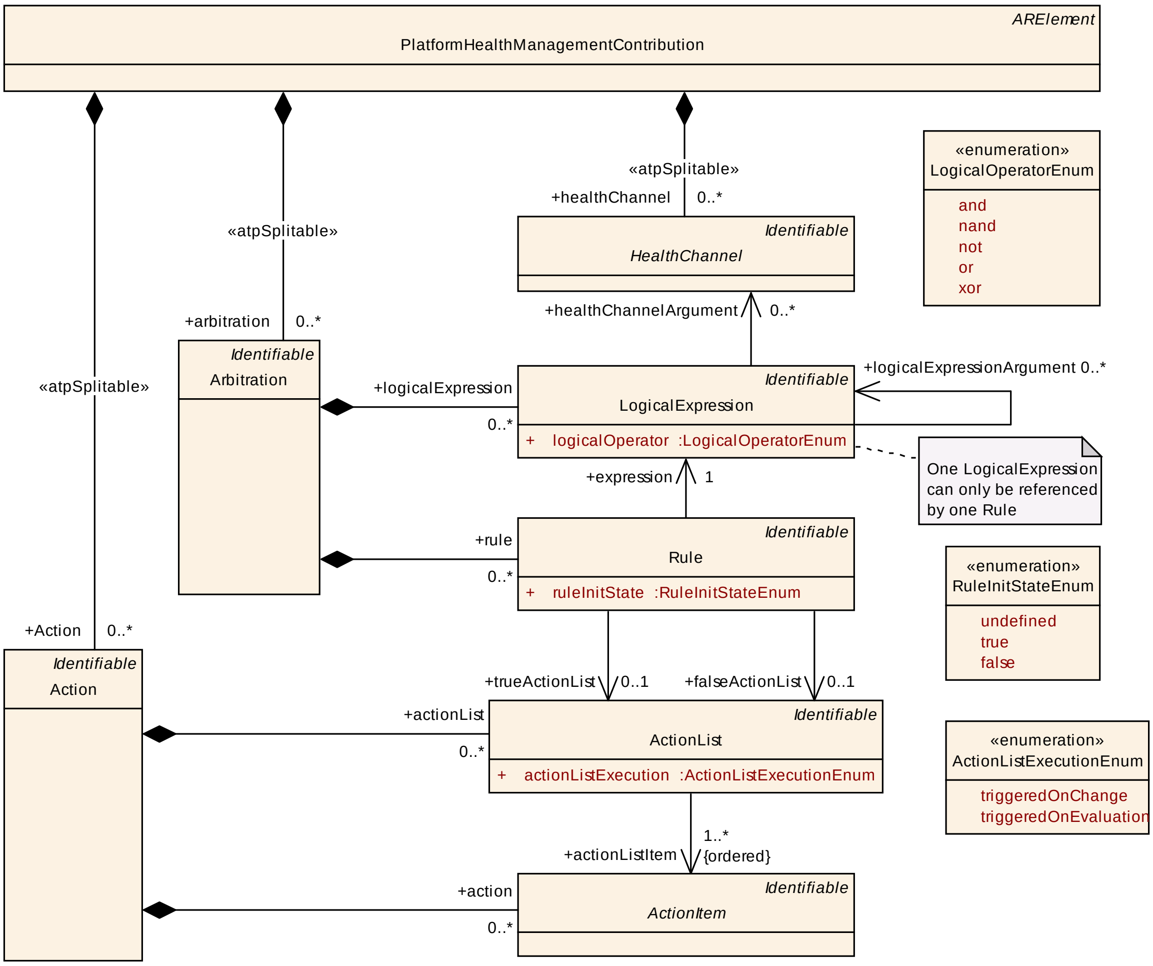

Figure 14.5: Modeling of Arbitration

[TPS_MANI_03518] LogicalExpression definition ⌈ A LogicalExpression 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 LogicalExpression can again be used as the input to another LogicalExpression.

There are some concerns which need to be formalized at a later point in time to make the definition of LogicalExpressions unambiguous:

- using more that 2 inputs for a LogicalExpression may lead to ambiguous definitions for some logicalOperators

- the inputs to the LogicalExpression are not ordered
- cyclic or recursive definition of LogicalExpressions have to be excluded
- ...

Class	**Arbitration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines logical expressions and rules to be evaluated by the platform health management. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
logicalExpression	LogicalExpression	*	aggr	Collection of LogicalExpressions in the context of an Arbitration. **Tags:** atp.Status=draft
rule	Rule	*	aggr	Collection of rules in the context of an Arbitration. **Tags:** atp.Status=draft

Table 14.15: Arbitration

Class	**LogicalExpression**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines a logical expression with an arbitrary number of arguments. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
healthChannelArgument	HealthChannel	*	ref	Reference to the HealthChannels which shall be considered for the evaluation of the LogicalExpression. **Tags:** atp.Status=draft
logicalExpressionArgument	LogicalExpression	*	ref	Reference to another LogicalExpression which shall be considered in the evaluation of this LogicalExpression. **Tags:** atp.Status=draft
logicalOperator	LogicalOperatorEnum	1	attr	Definition of the operator to be applied to this LogicalExpression. **Tags:** atp.Status=draft

Table 14.16: LogicalExpression

Enumeration	**LogicalOperatorEnum**

Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::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 14.17: LogicalOperatorEnum

The result of a LogicalExpression is taken as input to a Rule where it is decided whether and which reaction has to be performed.

[TPS_MANI_03519] Rule definition ⌈ A Rule takes the result of exactly one LogicalExpression and defines the handling of a reaction based on the result of the LogicalExpression:

- if the LogicalExpression evaluates to *true* the ActionList referenced in the role trueActionList will be indicated for execution
- if the LogicalExpression evaluates to *false* the ActionList referenced in the role falseActionList will be indicated for execution

Whether an ActionList is actually executed is depending on the setting of actionListExecution (see [TPS_MANI_03520]). ⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3527] LogicalExpression referenced by one Rule ⌈ Each LogicalExpression shall only be referenced by up to one Rule in the role Rule.expression. ⌋*()*

Class	**Rule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines a rule for the platform health management. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
expression	LogicalExpression	1	ref	Reference to the logical expression that is evaluated for this rule. **Tags:** atp.Status=draft
falseActionList	ActionList	0..1	ref	Reference to the action list which shall be executed when the rule evaluates to FALSE. **Tags:** atp.Status=draft

ruleInitState	RuleInitStateEnum	1	attr	Defines the initial state of this rule. **Tags:** atp.Status=draft
trueActionList	ActionList	0..1	ref	Reference to the action list which shall be executed when the rule evaluates to TRUE. **Tags:** atp.Status=draft

Table 14.18: Rule

Enumeration	**RuleInitStateEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement
Note	Definition of initial states for rules. **Tags:** atp.Status=draft
Literal	***Description***
false	**Tags:** atp.EnumerationValue=2
true	**Tags:** atp.EnumerationValue=1
undefined	**Tags:** atp.EnumerationValue=0

Table 14.19: RuleInitStateEnum

The ActionList collects an ordered list of ActionItems to be executed when the ActionList is executed. Whether an ActionList is actually executed is defined by the actionListExecution.

[TPS_MANI_03520] Execution of ActionList with actionListExecution=triggeredOnEvaluation ⌈ When a Rule indicates the execution of an ActionList with actionListExecution=triggeredOnEvaluation this ActionList is unconditionally executed every time the Rule is evaluated. ⌋ *(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03521] Execution of ActionList with actionListExecution=triggeredOnChange ⌈ When a Rule indicates the execution of an ActionList with actionListExecution=triggeredOnChange this ActionList is only executed when the previous state of the Rule was different from the current state. ⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**ActionList**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement			
Note	This element defines an action list for the platform health management. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

actionListExecution	ActionListExecutionEnum	1	attr	Defines the execution semantics for this action list. **Tags:** atp.Status=draft
actionListItem (ordered)	ActionItem	1..*	ref	Ordered reference to the action items to be executed in the scope of this action list. **Tags:** atp.Status=draft

Table 14.20: ActionList

Enumeration	**ActionListExecutionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement
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
triggeredOnEvaluation	Actions shall be executed every time the evaluation of the corresponding rule is done. **Tags:** atp.EnumerationValue=1

Table 14.21: ActionListExecutionEnum

Class	**ActionItem (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement			
Note	This element defines one possible action for the platform health management. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 14.22: ActionItem

14.6 Action deployment

Actions are executed in the scope of an `ActionList` in a well defined order. The specific subtypes of actions are described below.

14.6.1 Application action deployment

The ApplicationActionItem 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 [15] by the Platform Health Management.

[TPS_MANI_03522] Definition of actions for application software ⌈The ApplicationActionItem defines an action for a specific Process. The action can be either to terminate or to restart the Process.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**ApplicationActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines the action to be performed for one specific application instance. **Tags:** atp.Status=draft			
Base	ARObject, ActionItem, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
actionType	ApplicationActionTypeEnum	1	attr	Defines the action be performed on this application instance. **Tags:** atp.Status=draft
process	Process	0..1	ref	Reference to the process which represents the application instance. **Tags:** atp.Status=draft

Table 14.23: ApplicationActionItem

Enumeration	**ApplicationActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management
Note	Definition of available actions to be applied to an application instance. **Tags:** atp.Status=draft
Literal	*Description*
restart	**Tags:** atp.EnumerationValue=1
terminate	**Tags:** atp.EnumerationValue=0

Table 14.24: ApplicationActionTypeEnum

14.6.2 Platform action deployment

The PlatformActionItem defines an action which is targeting the whole Platform Instance. The action will be forwarded to the Execution Management [15] by the Platform Health Management.

[TPS_MANI_03523] Definition of actions for Platform Instance ⌈ The PlatformActionItem defines an action for the Platform Instance. Different kinds of possible reset strategies are defined in the attribute PlatformActionItem.actionType. ⌋ *(RS_MANI_00023, RS_MANI_00032)*

Class	**PlatformActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement			
Note	This element defines the action to be performed for this platform instance. **Tags:** atp.Status=draft			
Base	ARObject, ActionItem, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
actionType	PlatformActionTypeEnum	1	attr	Defines the action be performed on this platform instance. **Tags:** atp.Status=draft

Table 14.25: PlatformActionItem

Enumeration	**PlatformActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement
Note	Definition of available actions to be applied to an application instance. **Tags:** atp.Status=draft
Literal	*Description*
resetEcu	**Tags:** atp.EnumerationValue=2
resetMcu	**Tags:** atp.EnumerationValue=1
resetVm	**Tags:** atp.EnumerationValue=0

Table 14.26: PlatformActionTypeEnum

14.6.3 Watchdog action deployment

The WatchdogActionItem defines an action which is specific to a Watchdog.

[TPS_MANI_03524] Definition of actions for Watchdog ⌈ The WatchdogActionItem defines an action for the Watchdog. Currently only one WatchdogActionItem.actionType for the watchdog is defined, namely to stop triggering of the watchdog. ⌋ *(RS_MANI_00023, RS_MANI_00032)*

Class	WatchdogActionItem			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management			
Note	This element defines the action be performed on the watchdog. **Tags:** atp.Status=draft			
Base	ARObject, ActionItem, Identifiable, MultilanguageReferrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
actionType	WatchdogActionTypeEnum	1	attr	Defines the action to be performed on the watchdog. **Tags:** atp.Status=draft

Table 14.27: WatchdogActionItem

Enumeration	WatchdogActionTypeEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealth Management
Note	Definition of available actions to be applied to a watchdog. **Tags:** atp.Status=draft
Literal	**Description**
stopTrigger	**Tags:** atp.EnumerationValue=0

Table 14.28: WatchdogActionTypeEnum

15 Uploadable Software Package

15.1 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 SoftwareClusterRequirements.

The relation between SoftwareClusterRequirement and SoftwareCluster is depicted in Figure 15.1.

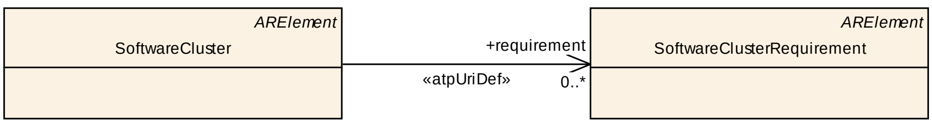

Figure 15.1: Relation of SoftwareClusterRequirement to SoftwareCluster

[TPS_MANI_01109] 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 SoftwareClusterRequirement 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			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table 15.1: UploadablePackageElement

Please note that this approach to collecting elements is very similar in semantics to System.fibexElement or DiagnosticContributionSet.element.

15.2 Software Cluster Requirement

[TPS_MANI_01112] Semantics of SoftwareClusterRequirement ⌈ The existence of a SoftwareClusterRequirement 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 SoftwareClusterRequirement 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 SoftwareClusterRequirement 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 SoftwareClusterRequirement is motivated from the methodological point of view.

[TPS_MANI_01113] Semantics of SoftwareClusterRequirement.diagnosticAddress ⌈ The existence of the attribute SoftwareClusterRequirement.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)*

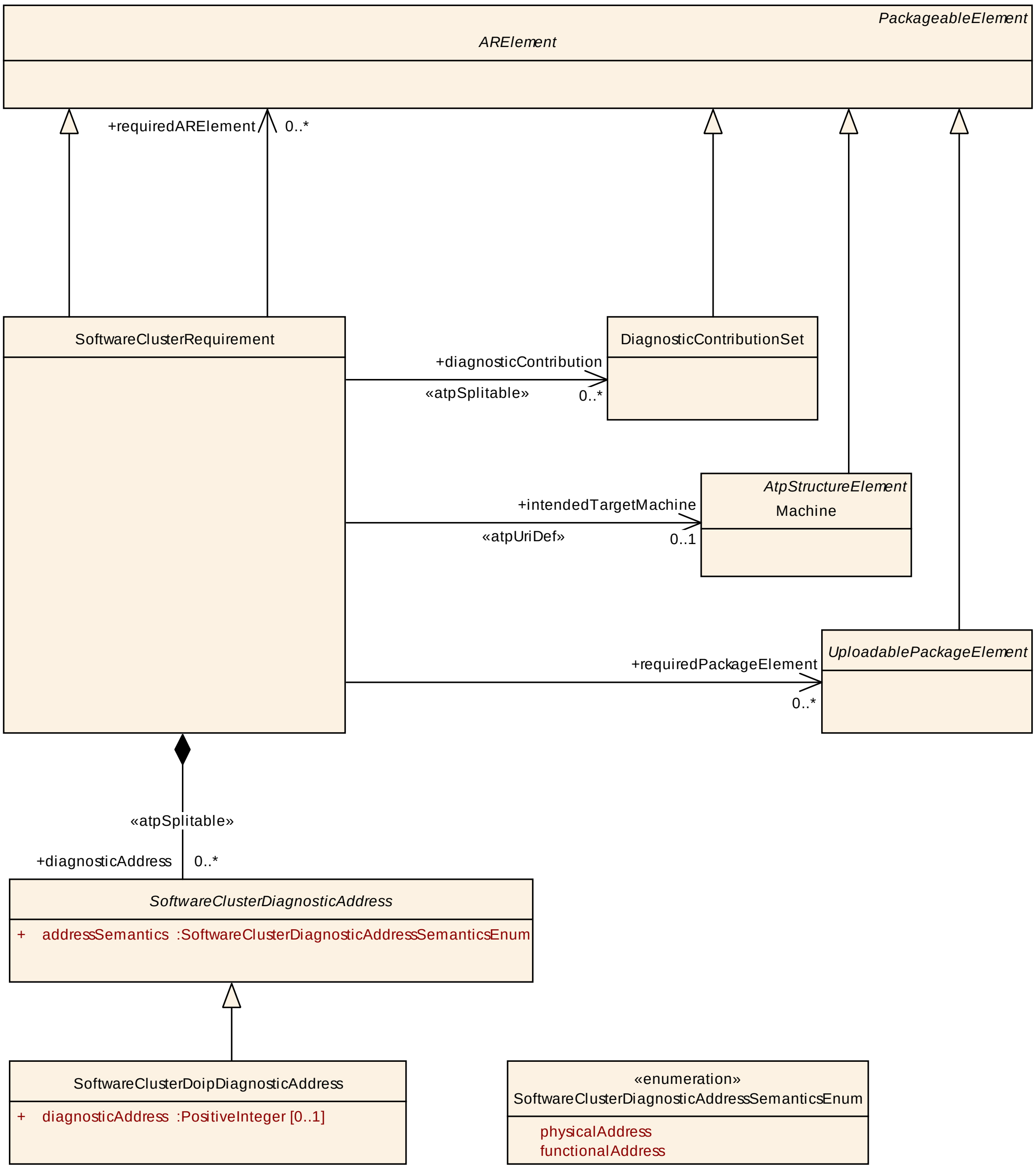

Figure 15.2: Modeling of SoftwareClusterRequirement

Class	**SoftwareClusterRequirement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareCluster			
Note	This meta-class represents the ability for the OEM to specify the grouping of software uploadable to a specific target Machine. **Tags:** atp.Status=draft; atp.recommendedPackage=SoftwareClusterRequirements			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

diagnosticAddress	SoftwareClusterDiagnosticAddress	*	aggr	This aggregaton is used to specify the diagnsotic address. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticAddress; atp.Status=draft
diagnosticContribution	DiagnosticContributionSet	*	ref	This reference identifes the corresponding collection of DiagnosticContributionSet. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft
intendedTargetMachine	Machine	0..1	ref	This reference can be taken to identify the machine for which the final SoftwareCluster shall be developed. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
requiredARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackageElement and that are required for the completeness of the definition of the SoftwareCluster. **Tags:** atp.Status=draft
requiredPackageElement	UploadablePackageElement	*	ref	This reference points to uploadable elements that have been identified as relevant in the context of the enclosing SoftwareClusterRequirement. **Tags:** atp.Status=draft

Table 15.2: SoftwareClusterRequirement

[TPS_MANI_01117] Semantics of SoftwareClusterRequirement.intendedTargetMachine ⌈ The specification of SoftwareClusterRequirement.intendedTargetMachine allows for focusing the specification of an uploadable software package to a specific Machine 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] Relation between SoftwareClusterRequirement and DiagnosticContributionSet ⌈ An important aspect of the definition of a SoftwareClusterRequirement is the question what diagnostic extract shall be associated with the SoftwareClusterRequirement.

For this purpose, a reference from SoftwareClusterRequirement 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)*

Please mind the intentionally introduced difference between SoftwareCluster and SoftwareClusterRequirement in terms of the relation to DiagnosticContributionSet.

In other words, the multiplicity of the references to DiagnosticContributionSet intentionally differ.

As already explained, the SoftwareClusterRequirement shall support the decentralized configuration of the DiagnosticContributionSet while the SoftwareCluster requires the existence of a final (merged) DiagnosticContributionSet.

[TPS_MANI_01119] Reference to model elements from SoftwareClusterRequirement ⌈ SoftwareClusterRequirement 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 SoftwareCluster.containedPackageElement.
- references to meta-classes derived from ARElement are formalized by way of SoftwareCluster.containedARElement.

⌋*(RS_MANI_00035)*

Please note that the conversion of a SoftwareClusterRequirement 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 SoftwareClusterRequirement around for a longer period in time.

15.3 Software Cluster

[TPS_MANI_01110] Semantics of SoftwareCluster ⌈ The existence of a SoftwareCluster represents an uploadable software package. ⌋*(RS_MANI_00035)*

[TPS_MANI_01111] 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*)

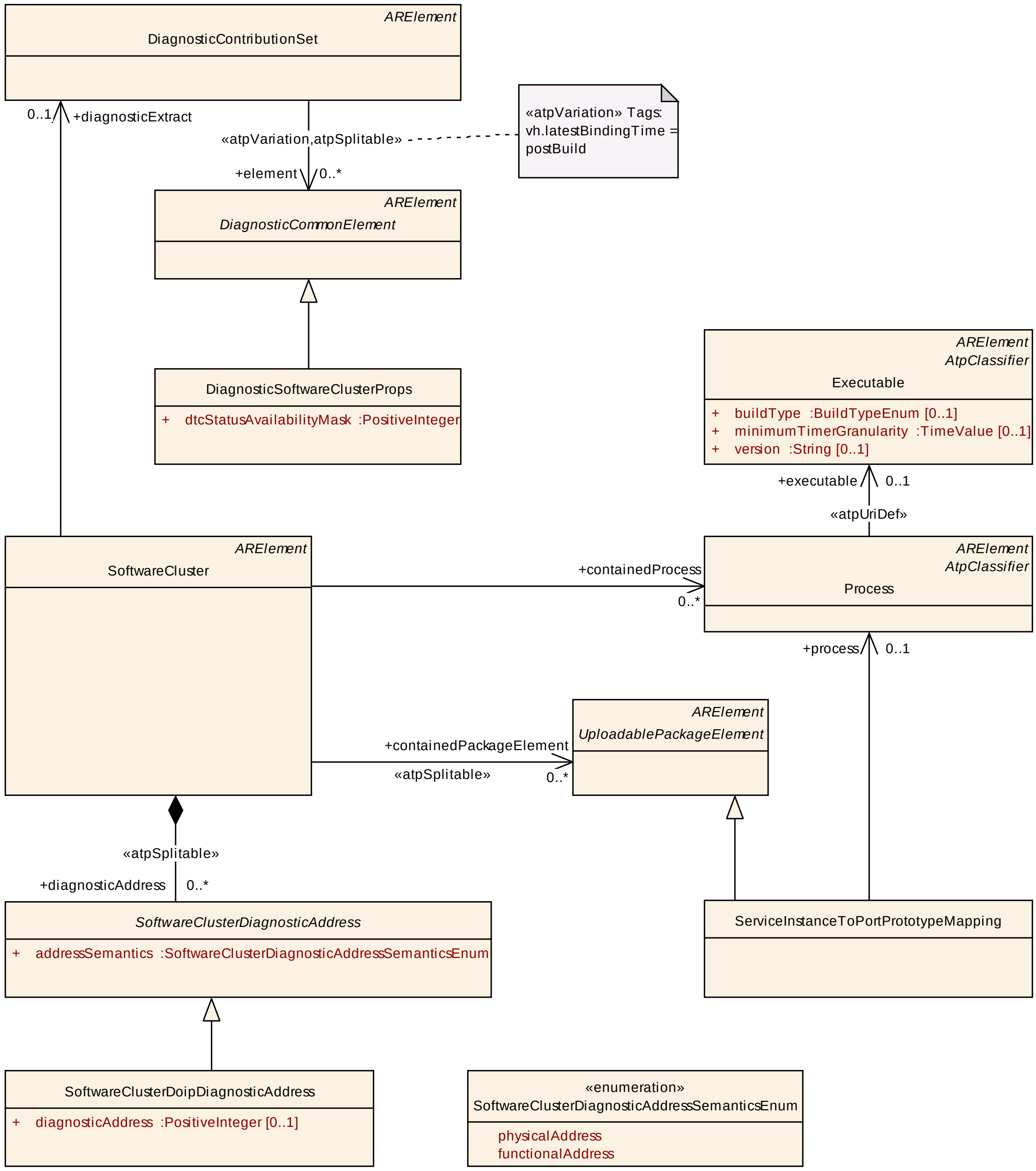

Figure 15.3: Modeling of SoftwareCluster

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.

[constr_1543] 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::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, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
containedARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackageElement and that contribute to the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft
containedPackageElement	UploadablePackageElement	*	ref	This reference identifies model elements that are required to complete the manifest content. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft
containedProcess	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster. **Tags:** atp.Status=draft
diagnosticAddress	SoftwareClusterDiagnosticAddress	*	aggr	This aggregation represents the collection of diagnostic addresses that apply for the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticAddress; atp.Status=draft
diagnosticExtract	DiagnosticContributionSet	0..1	ref	This reference represents the definition of the diagnostic extract applicable to the referencing SoftwareCluster **Tags:** atp.Status=draft

requirement	SoftwareCluster Requirement	*	ref	This reference represents the identifcaction of all requirements applicable for the enclosing software cluster. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 15.3: SoftwareCluster

Class	**SoftwareClusterDiagnosticAddress (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareCluster			
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			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
addressSemantics	SoftwareClusterDiagnosticAddressSemanticsEnum	1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address.

Table 15.4: SoftwareClusterDiagnosticAddress

Enumeration	**SoftwareClusterDiagnosticAddressSemanticsEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareCluster
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 15.5: SoftwareClusterDiagnosticAddressSemanticsEnum

Class	SoftwareClusterDoipDiagnosticAddress			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareCluster			
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 15.6: SoftwareClusterDoipDiagnosticAddress

[TPS_MANI_01114] 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 `element`s 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_1534] Existence of `DiagnosticSoftwareClusterProps` ⌈ Each `DiagnosticContributionSet` shall only reference one and only one `DiagnosticSoftwareClusterProps` in the role `element`. ⌋*()*

Class	**DiagnosticSoftwareClusterProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareCluster			
Note	This meta-class represents the ability to specify properties for the relation between a DiagnosticContributionSet and a SoftwareCluster. **Tags:** atp.Status=draft; atp.recommendedPackage=DiagnosticSoftwareCluster Propss			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
dtcStatusAvailabilityMask	PositiveInteger	1	attr	This attribute contains the value of the DTC status availability mask. **Tags:** atp.Status=draft

Table 15.7: DiagnosticSoftwareClusterProps

[constr_1535] Existence of DiagnosticSoftwareClusterProps in the context of a DiagnosticContributionSet ⌈ Each DiagnosticContributionSet shall only reference a single DiagnosticSoftwareClusterProps in the role element. ⌋*()*

[TPS_MANI_01115] 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.

In contrast to the definition of an AdaptiveAutosarApplication, the definition of SoftwareCluster shall never include multiple Machines. This remarkable difference between SoftwareCluster and AdaptiveAutosarApplication is expressed in [constr_1536].

[constr_1536] 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. ⌋*()*

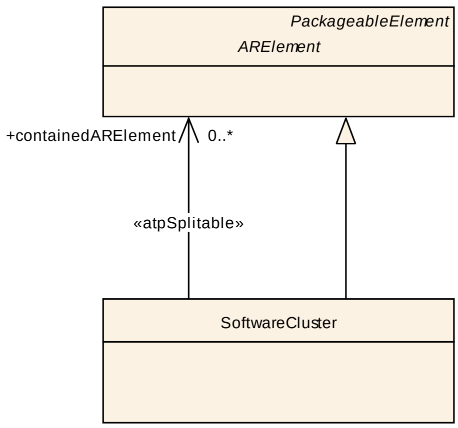

Figure 15.4: SoftwareCluster can reference ARElement

[TPS_MANI_01116] 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.

Technically, an UploadablePackageElement is also an ARElement, but it is still mandated to use the dedicated reference for UploadablePackageElement. ⌋ *(RS_MANI_00035)*

To exemplify the reference to UploadablePackageElement, Figure 15.3 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] No nested definition of SoftwareCluster ⌈ A SoftwareCluster shall not reference another SoftwareCluster in the role containedARElement. ⌋
()

16 REST Service Deployment

Important note: the AUTOSAR SWS REST [13] 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 need 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 application manifest) of the same service.

To further drive this point home, Figure 16.1 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.

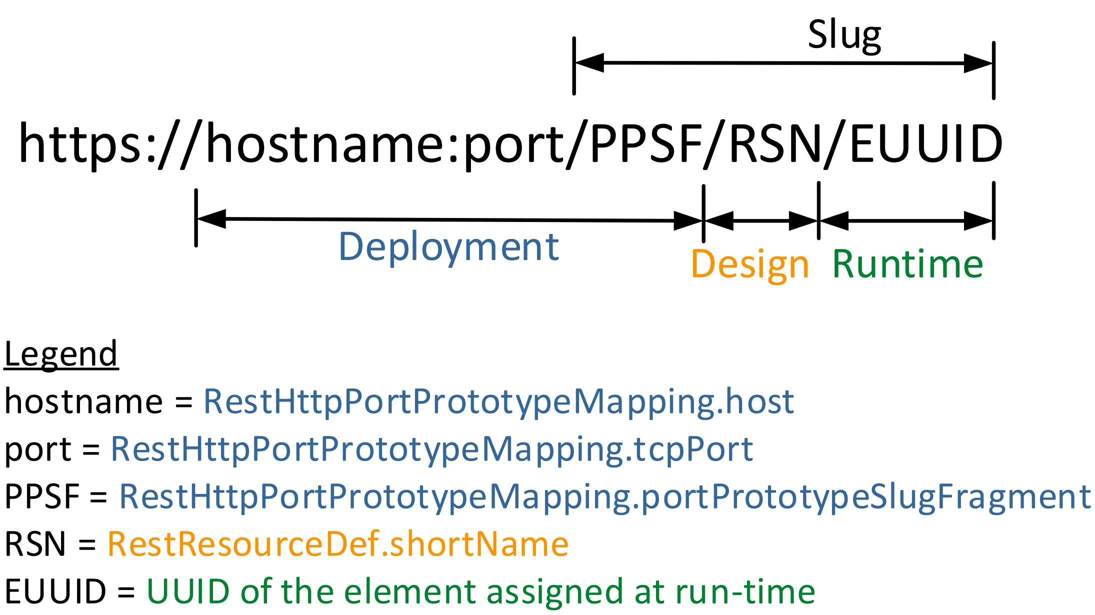

Figure 16.1: Structure of a typical URI for a REST service

As explained by Figure 16.1, 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 5. The contribution from the deployment phase is depicted in Figure 16.2.

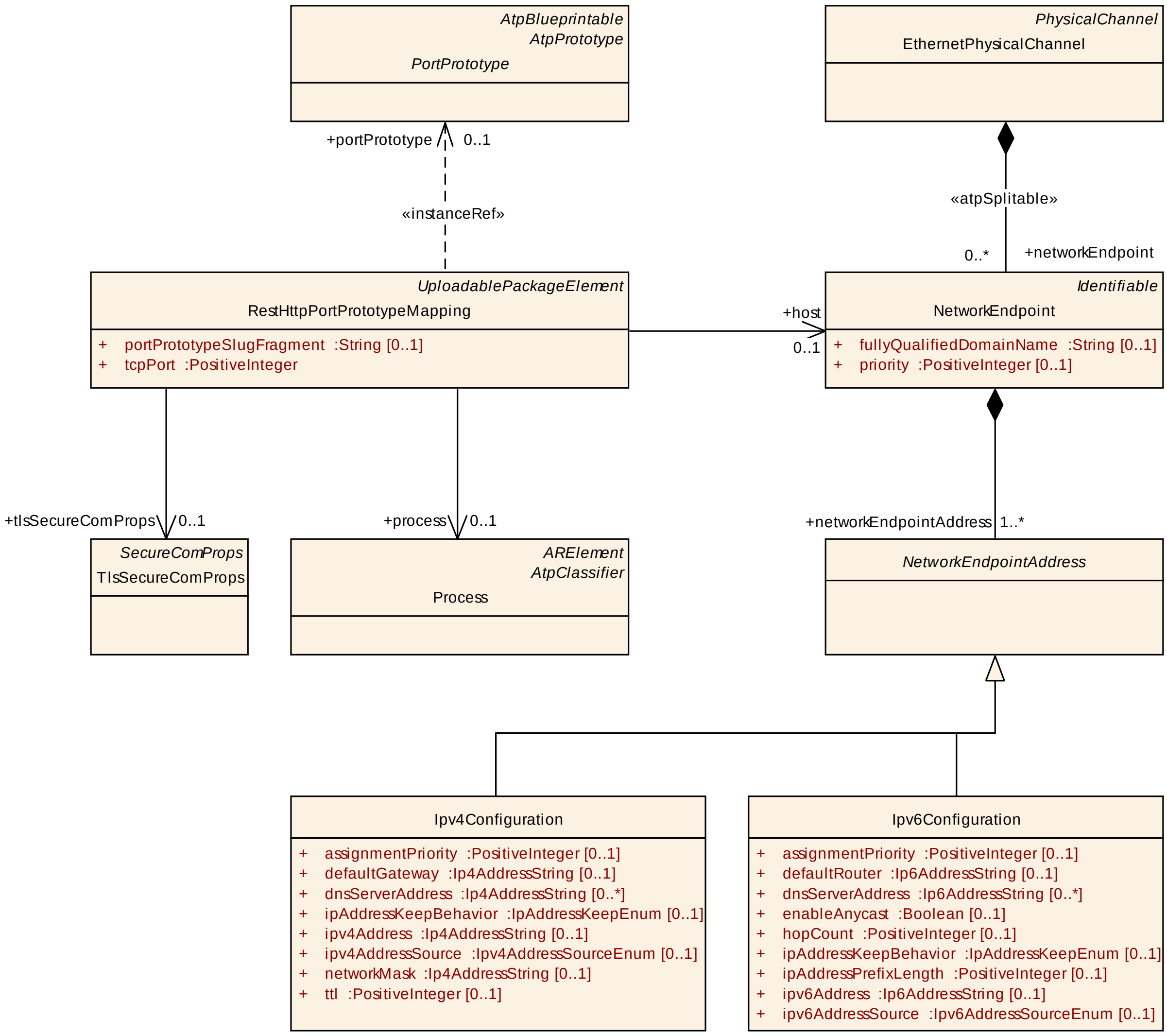

Figure 16.2: Modeling of the REST service deployment

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] 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 RestHttpPort-

PrototypeMapping.portPrototypeSlugFragment. This part is defined on deployment level in order to be sure that it is unique in the context to the *host-name: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] 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 16.1.

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 be 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::Deployment::REST			
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.Status=draft; atp.recommendedPackage=RestHttpPortPrototypeMappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

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 16.1: RestHttpPortPrototypeMapping

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.

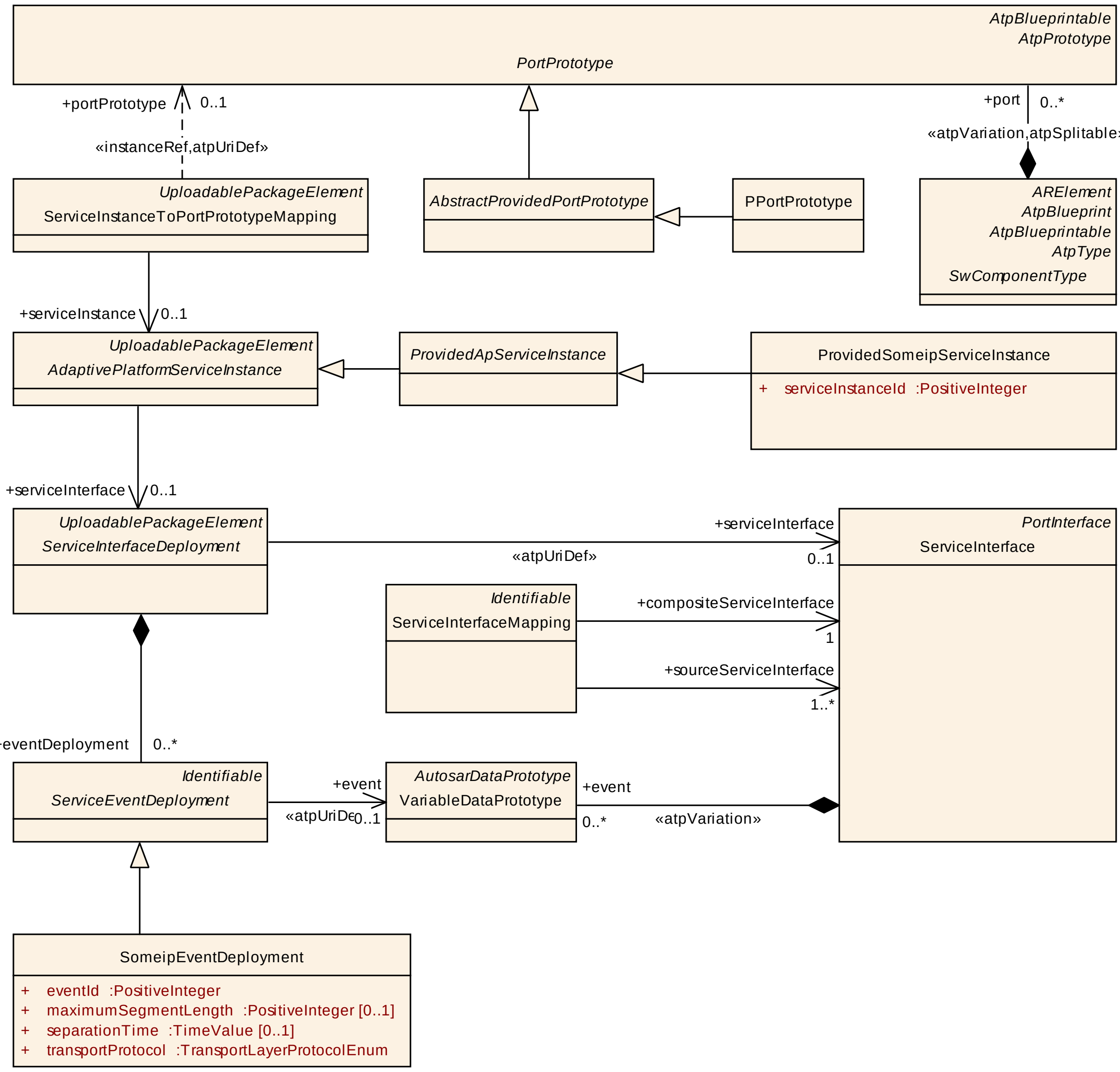

Figure A.1: Meta-model excerpt relevant for the example

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]).

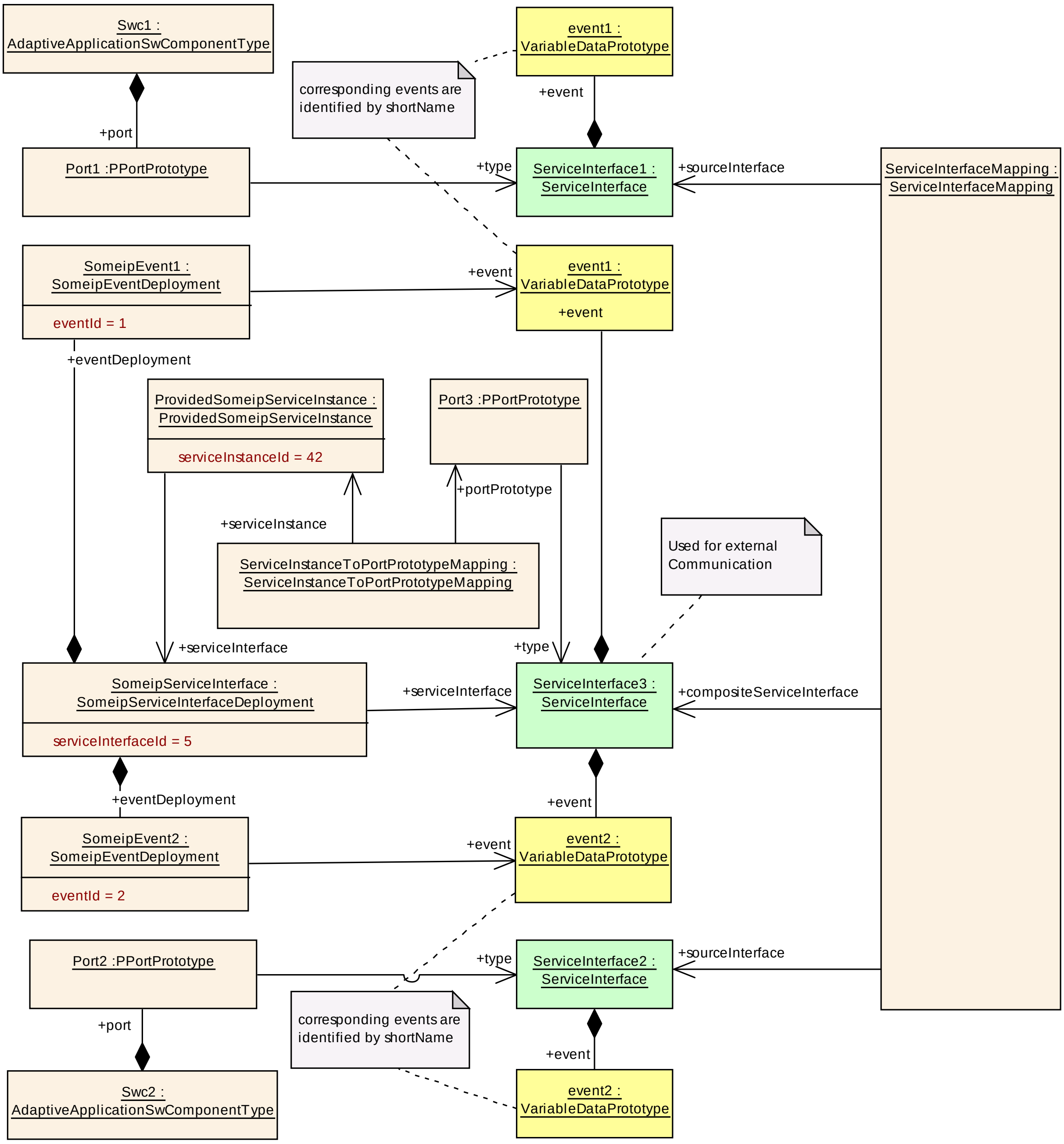

Figure A.2: Example for the deployments of a service in the presence of a ServiceInterfaceMapping

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 ServiceInterfaceEventMappings. 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.

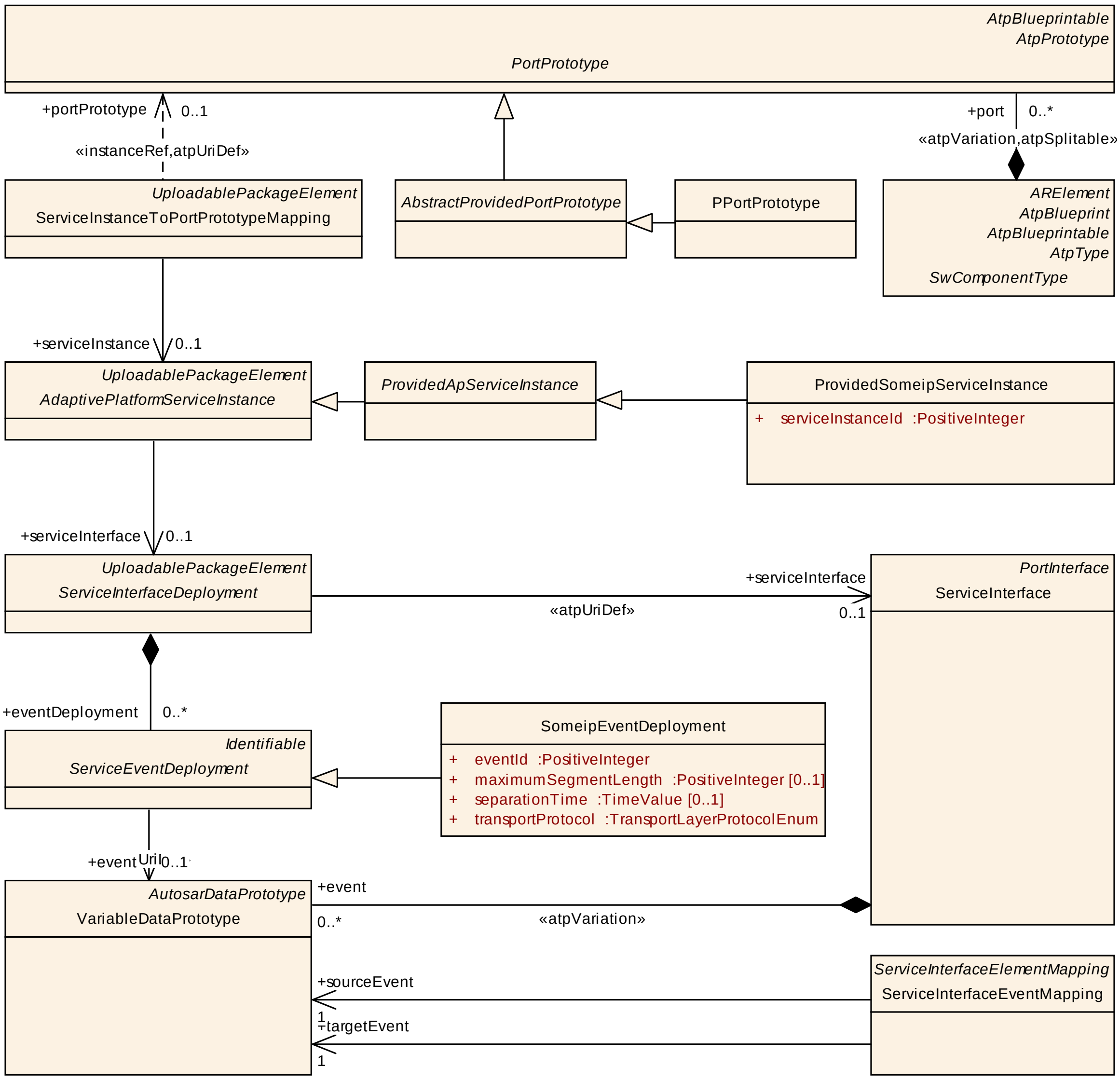

Figure A.3: Excerpt of the relevant meta-classes for the `ServiceInterfaceEventMapping` example

By mapping individual elements of `ServiceInterface`s, it is possible to map element with different `shortName`s to each other. In this example, the `event` with the `shortName` `event1` is mapped to another `event` with the `shortName` `eventLeft`.

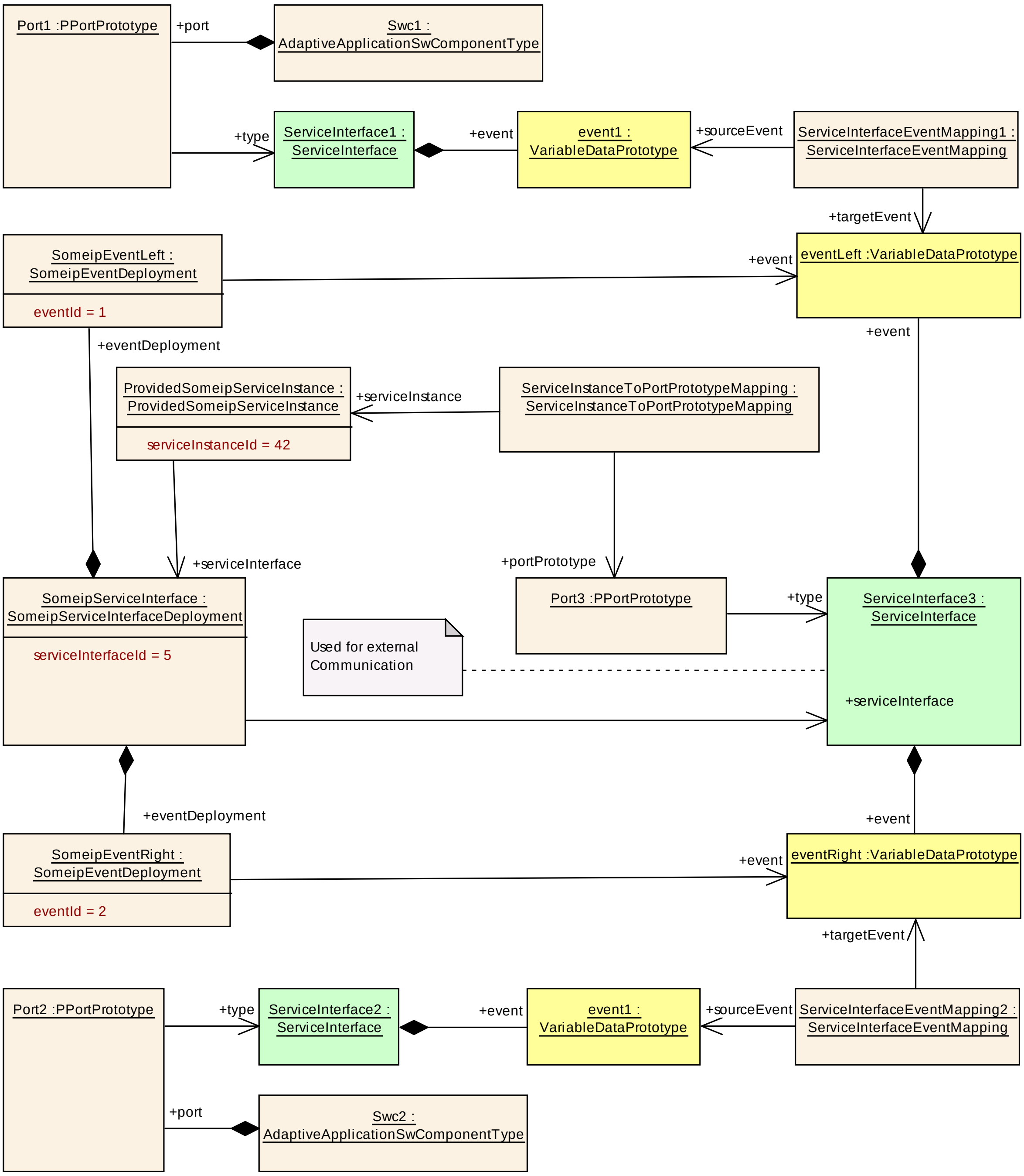

Figure A.4: Example for the deployment of a service in the presence of a ServiceInterfaceEventMapping

In Figure A.4, two different ServiceInterfaces exist that each aggregate an event with the identical shortName. This scenario **requires** the existence of ServiceInterfaceElementMappings.

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 ServiceInterfaceEventMappings that each refer to said event in the role ServiceInterfaceEventMapping.sourceEvent.

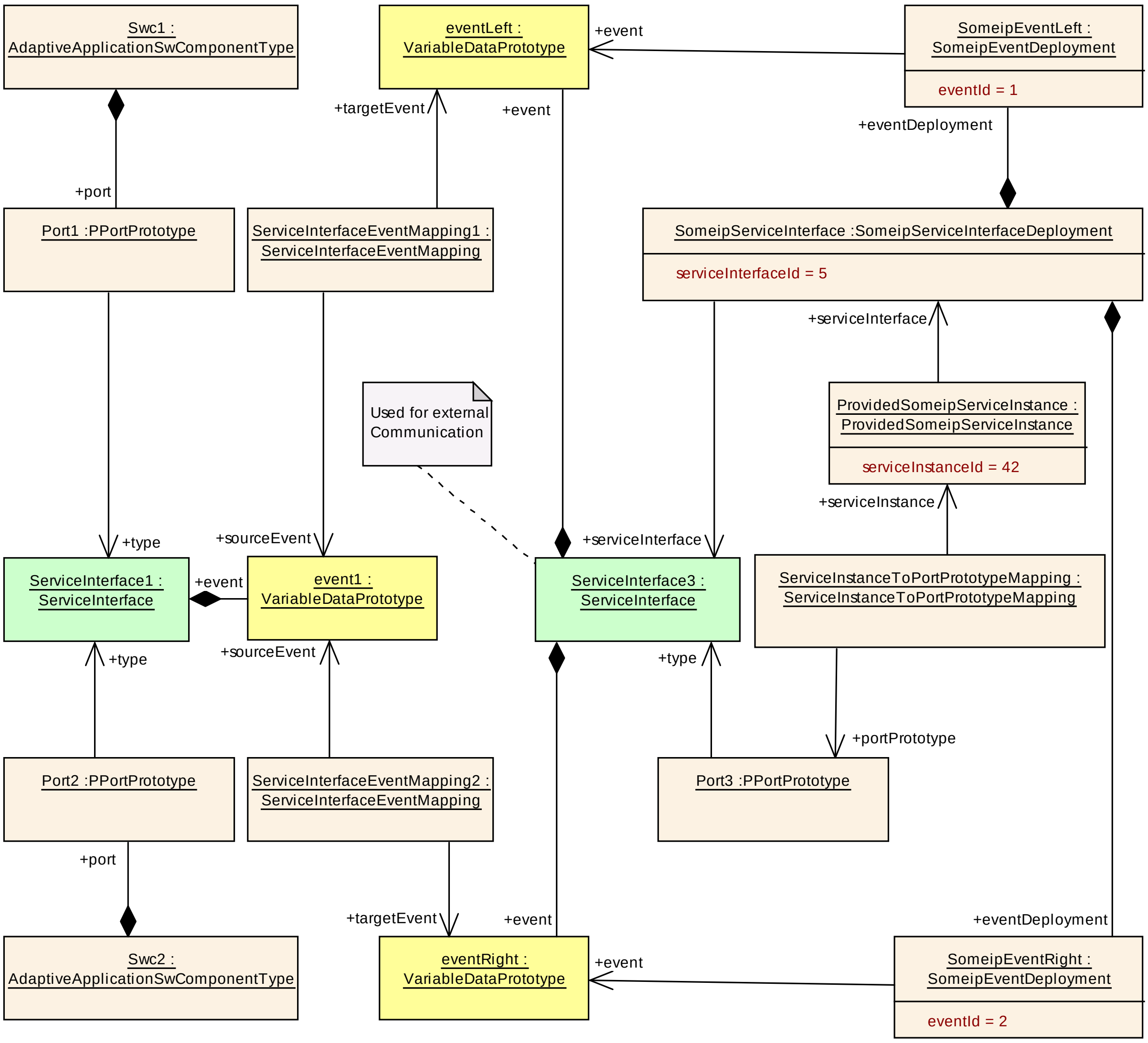

Figure A.5: Example for the deployment of a service in the presence of a ServiceInterfaceEventMapping to the same source ServiceInterface

Again, this scenario **requires** the existence of appropriately configured ServiceInterfaceElementMappings.

A.3 Definition of Startup Configuration

As already mentioned, the startup configuration is directly aggregated by the definition of a Process:

```
<PROCESS>
  <SHORT-NAME>AA1</SHORT-NAME>
  <MODE-DEPENDENT-STARTUP-CONFIGS>
    <MODE-DEPENDENT-STARTUP-CONFIG>
      <EXECUTION-DEPENDENCYS>
        <EXECUTION-DEPENDENCY>
          <APPLICATION-MODE-IREF>
```

```
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
DECLARATION-GROUP-PROTOTYPE">/Processes/MWC/ApplicationStateMachine</
CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
ModeDeclarationGroups/ApplicationStateMachine/Running</TARGET-MODE-
DECLARATION-REF>
        </APPLICATION-MODE-IREF>
      </EXECUTION-DEPENDENCY>
      <EXECUTION-DEPENDENCY>
        <APPLICATION-MODE-IREF>
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
DECLARATION-GROUP-PROTOTYPE">/Processes/MSM/ApplicationStateMachine</
CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
ModeDeclarationGroups/ApplicationStateMachine/Running</TARGET-MODE-
DECLARATION-REF>
        </APPLICATION-MODE-IREF>
      </EXECUTION-DEPENDENCY>
    </EXECUTION-DEPENDENCYS>
    <MACHINE-MODE-IREFS>
      <MACHINE-MODE-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>
      </MACHINE-MODE-IREF>
    </MACHINE-MODE-IREFS>
    <STARTUP-CONFIG-REF DEST="STARTUP-CONFIG">/StartupConfigSets/
StartupConfigSet_AA/AA1_Startup</STARTUP-CONFIG-REF>
  </MODE-DEPENDENT-STARTUP-CONFIG>
 </MODE-DEPENDENT-STARTUP-CONFIGS>
</PROCESS>
```

Listing A.1: Example for the definition of the ModeDependentStartupConfig owned by a Process

In this example, launch dependencies exist on two other Processes. Both Processes MWC and MSM need to be in the ApplicationState "Running" before AA1 is started.

The reference ModeDependentStartupConfig.machineMode refers to a ModeDeclaration with the shortName Driving within the state machine of the underlying Machine.

The referenced StartupConfig is defined in Listing A.2.

```
<STARTUP-CONFIG>
  <SHORT-NAME>AA1_Startup</SHORT-NAME>
  <RESOURCE-GROUP-REFS>
    <RESOURCE-GROUP-REF DEST="RESOURCE-GROUP">/Machines/ExampleMachine/
Linux/limitcpu</RESOURCE-GROUP-REF>
    <RESOURCE-GROUP-REF DEST="RESOURCE-GROUP">/Machines/ExampleMachine/
Linux/limitmem</RESOURCE-GROUP-REF>
  </RESOURCE-GROUP-REFS>
```

```xml
  <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 ResourceGroups named `limitcpu` and `limitmem`. This aspect can be found in Listing A.3.

```xml
<MACHINE>
  <SHORT-NAME>ExampleMachine</SHORT-NAME>
  <MACHINE-MODE-MACHINES>
    <MODE-DECLARATION-GROUP-PROTOTYPE>
      <SHORT-NAME>ExampleMachine_StateMachine</SHORT-NAME>
      <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/
   VehicleStateMachine</TYPE-TREF>
    </MODE-DECLARATION-GROUP-PROTOTYPE>
  </MACHINE-MODE-MACHINES>
  <MODULE-INSTANTIATIONS>
    <OS-MODULE-INSTANTIATION>
      <SHORT-NAME>Linux</SHORT-NAME>
      <RESOURCE-GROUPS>
        <RESOURCE-GROUP>
          <SHORT-NAME>limitcpu</SHORT-NAME>
          <DESC>
            <L-2 L="EN">Limits the cpu shares available to processes in
   this cgroup to 10.</L-2>
          </DESC>
        </RESOURCE-GROUP>
        <RESOURCE-GROUP>
          <SHORT-NAME>limitmem</SHORT-NAME>
          <DESC>
            <L-2 L="EN">Limits memory available to the cgroup processes to
   50MB. </L-2>
          </DESC>
        </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 resp. RequiredSomeipServiceInstance with the **identical value** of attribute serviceInstanceId creates the actual binding between the “find” and the “offer” end.

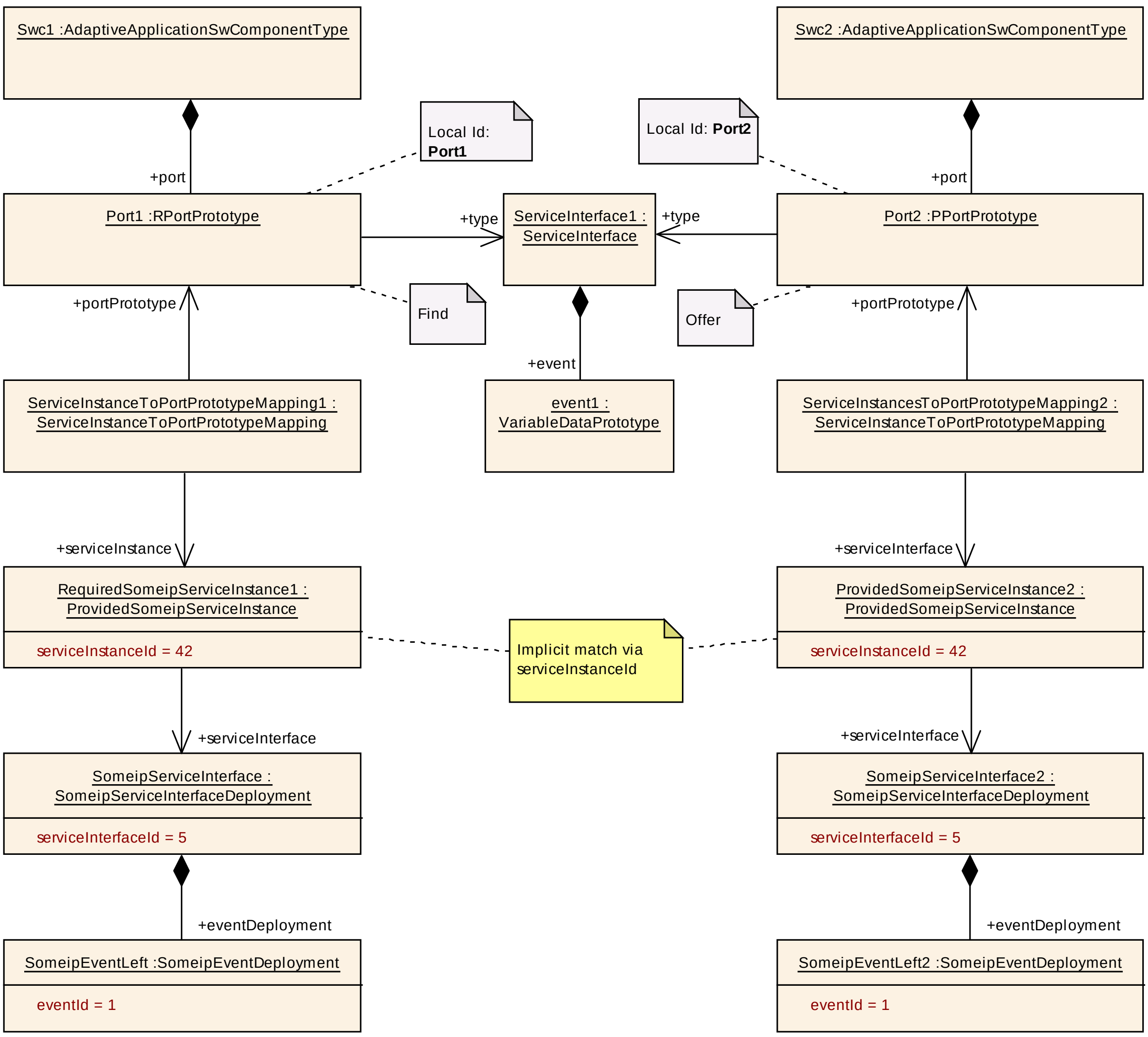

Figure A.6: Port-based binding of a service instance to the application using SOME/IP

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.

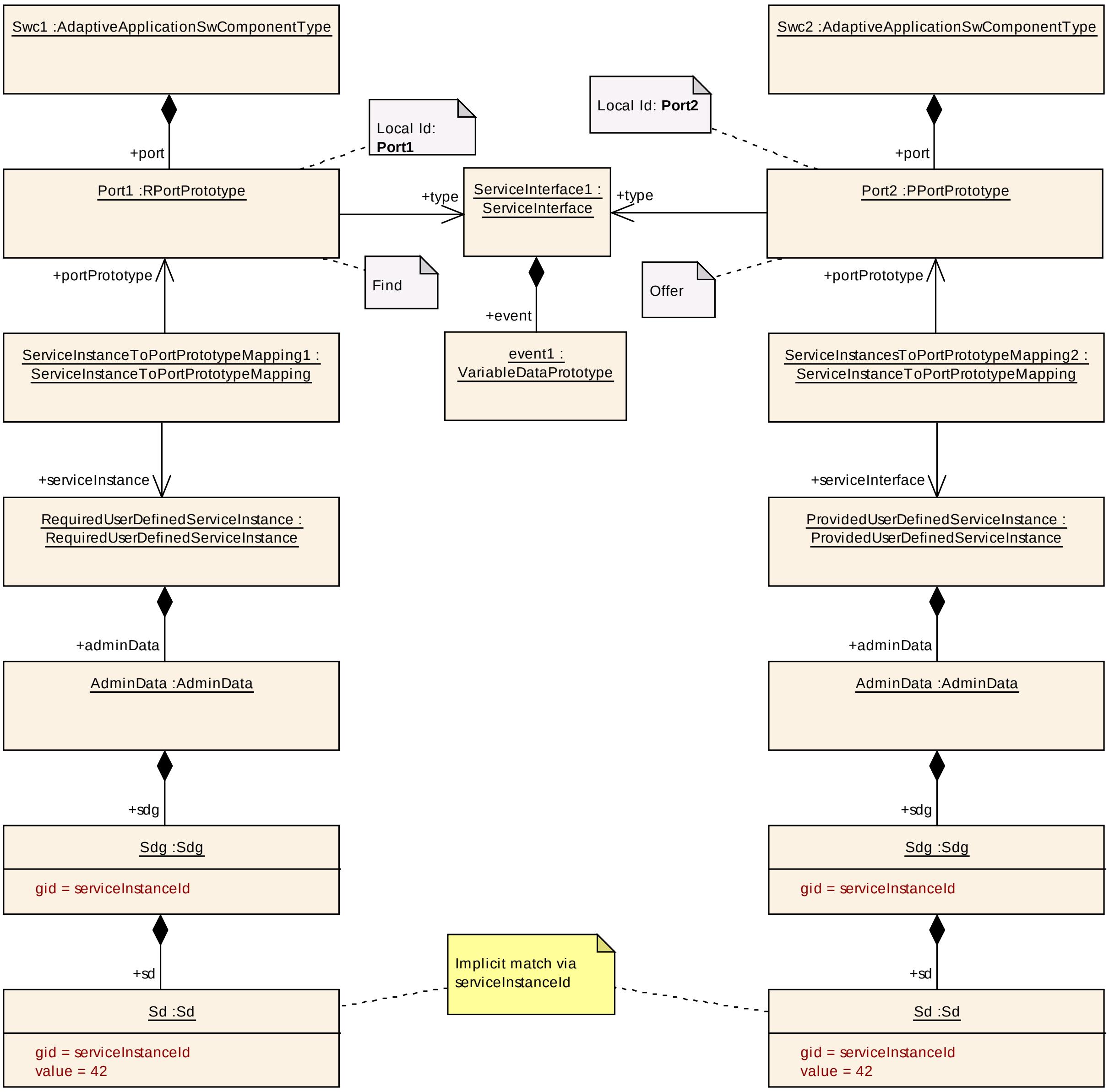

Figure A.7: Port-based binding of a service instance to the application using a user-defined binding

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`.

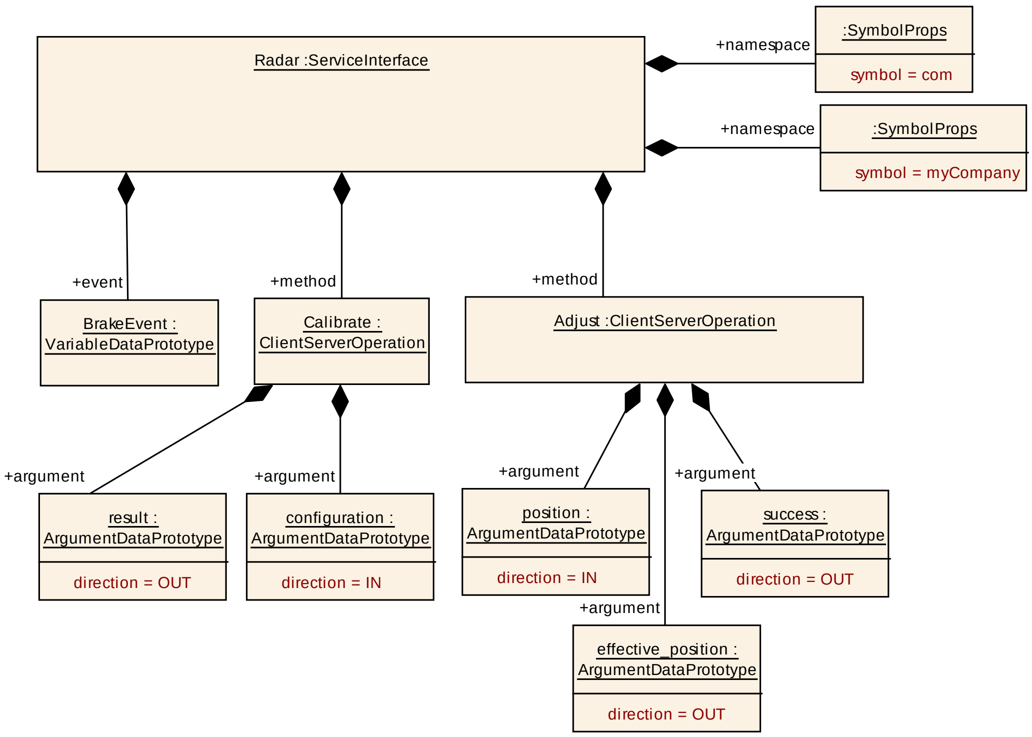

Figure A.8: Radar Service Interface

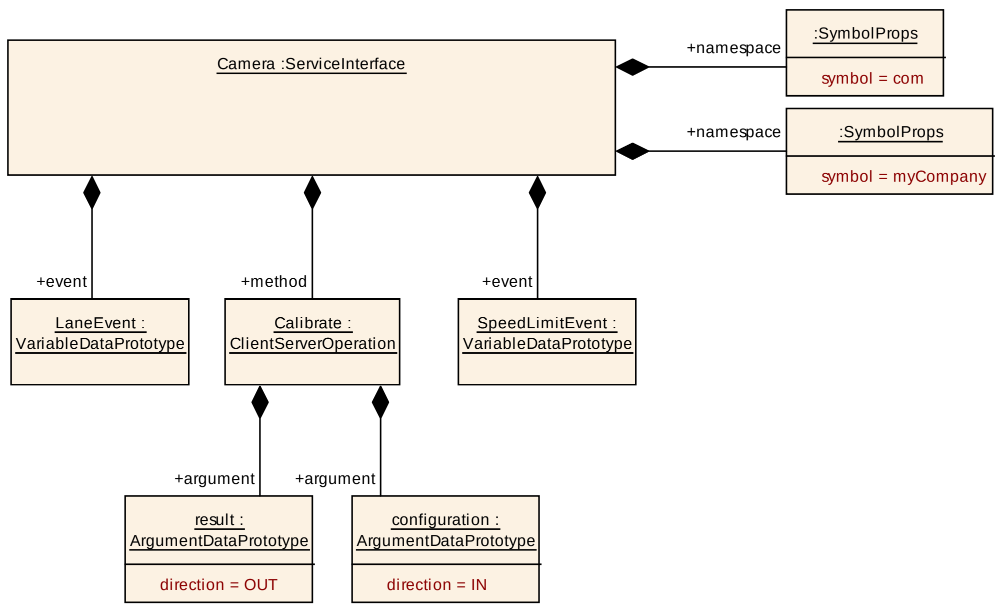

Figure A.9: Camera Service Interface

Both ServiceInterfaces *Radar* and *Camera* are mapped to a combined *RadarAndCamera* ServiceInterface with an Service Interface Element Mapping since both ServiceInterfaces have a method with the same name: *Calibrate*.

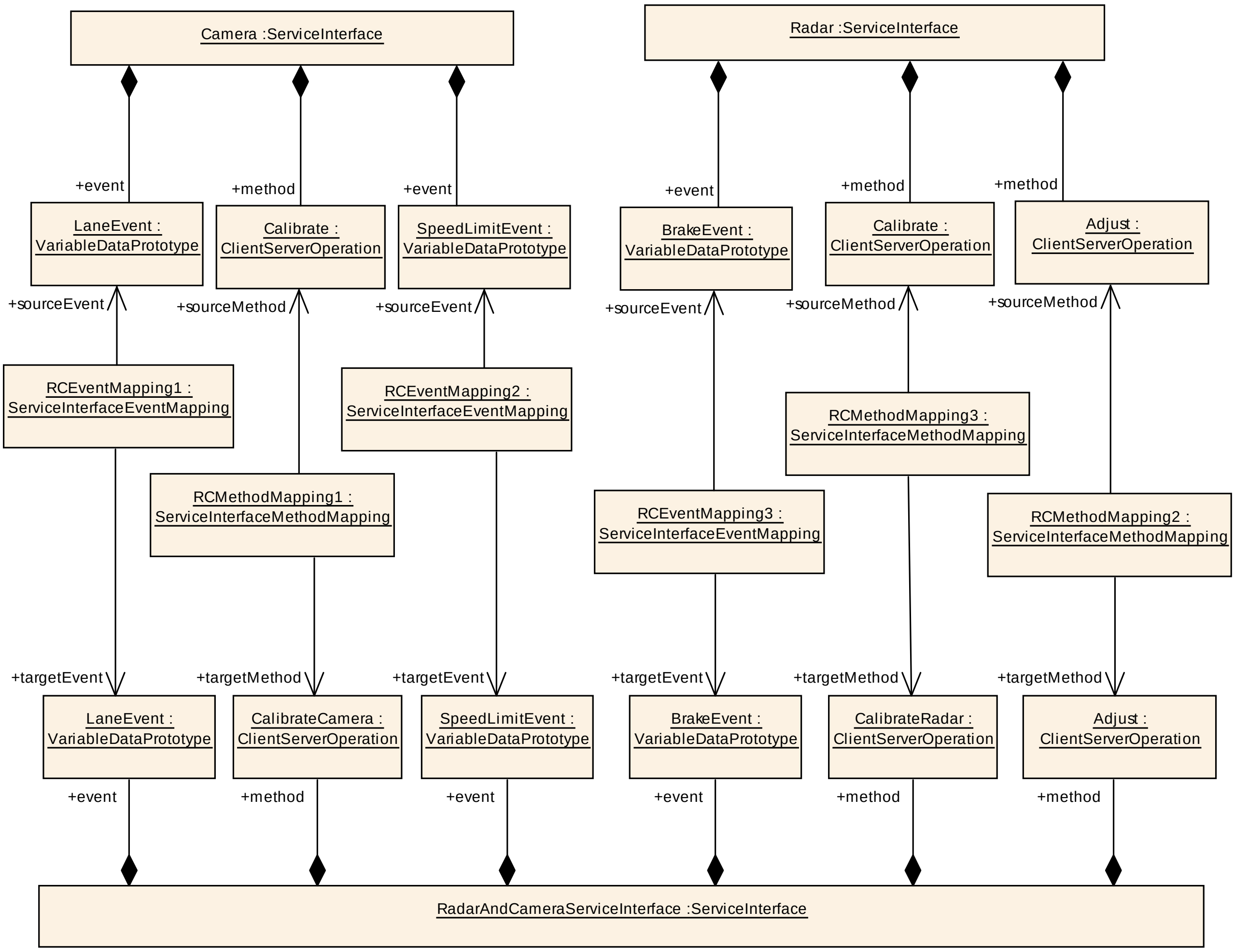

Figure A.10: Service Interface Element Mapping example

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 eventIds are assigned to the events and methodIds are assigned to the methods. Furthermore a single SomeipEventGroup is defined to which all SomeipEventDeployments of the *RadarAndCamera* ServiceInterface are assigned.

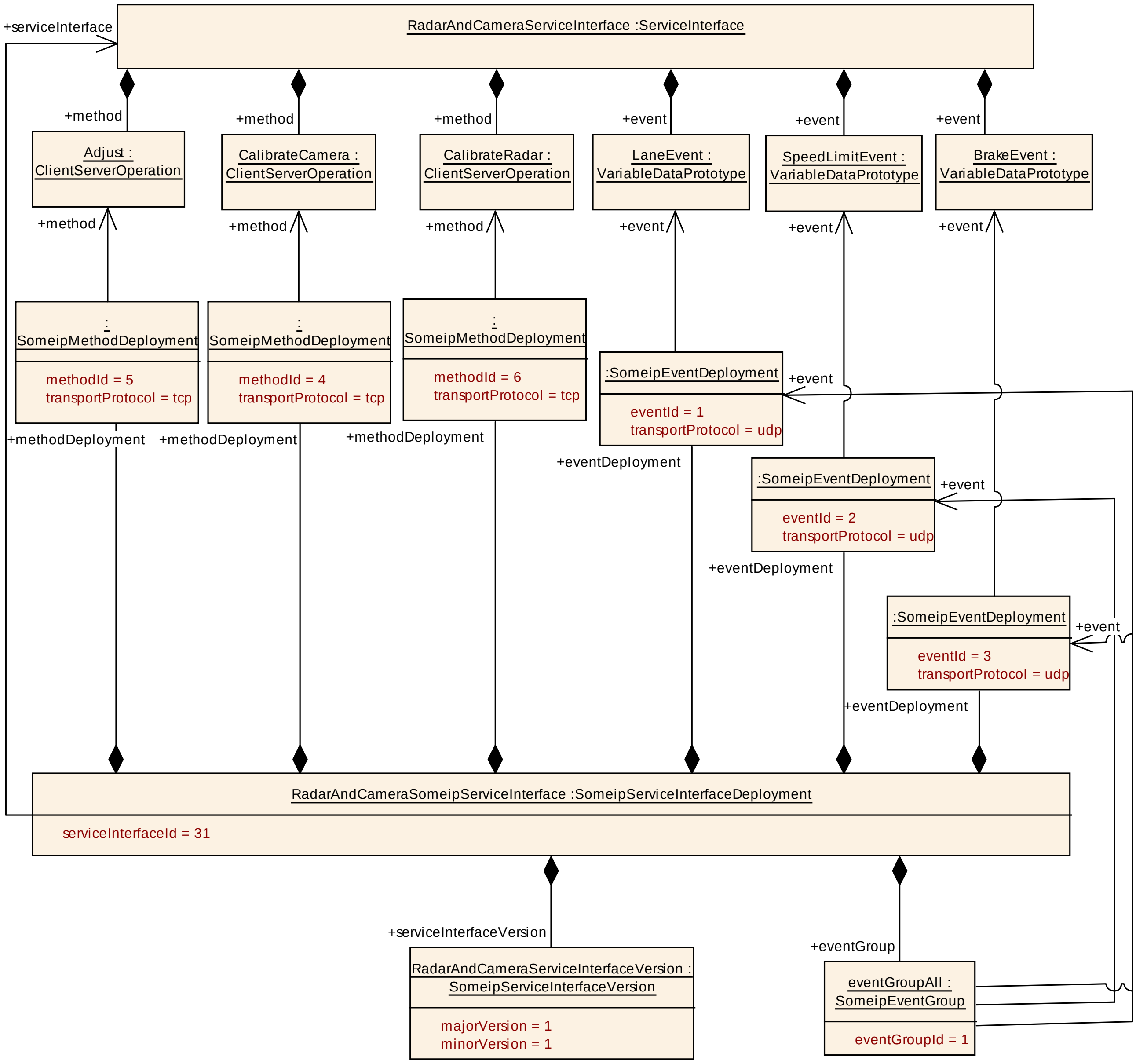

Figure A.11: SOME/IP Deployment

Figure A.12 shows a modeled `ProvidedSomeipServiceInstance` that is mapped to a `Machine`.

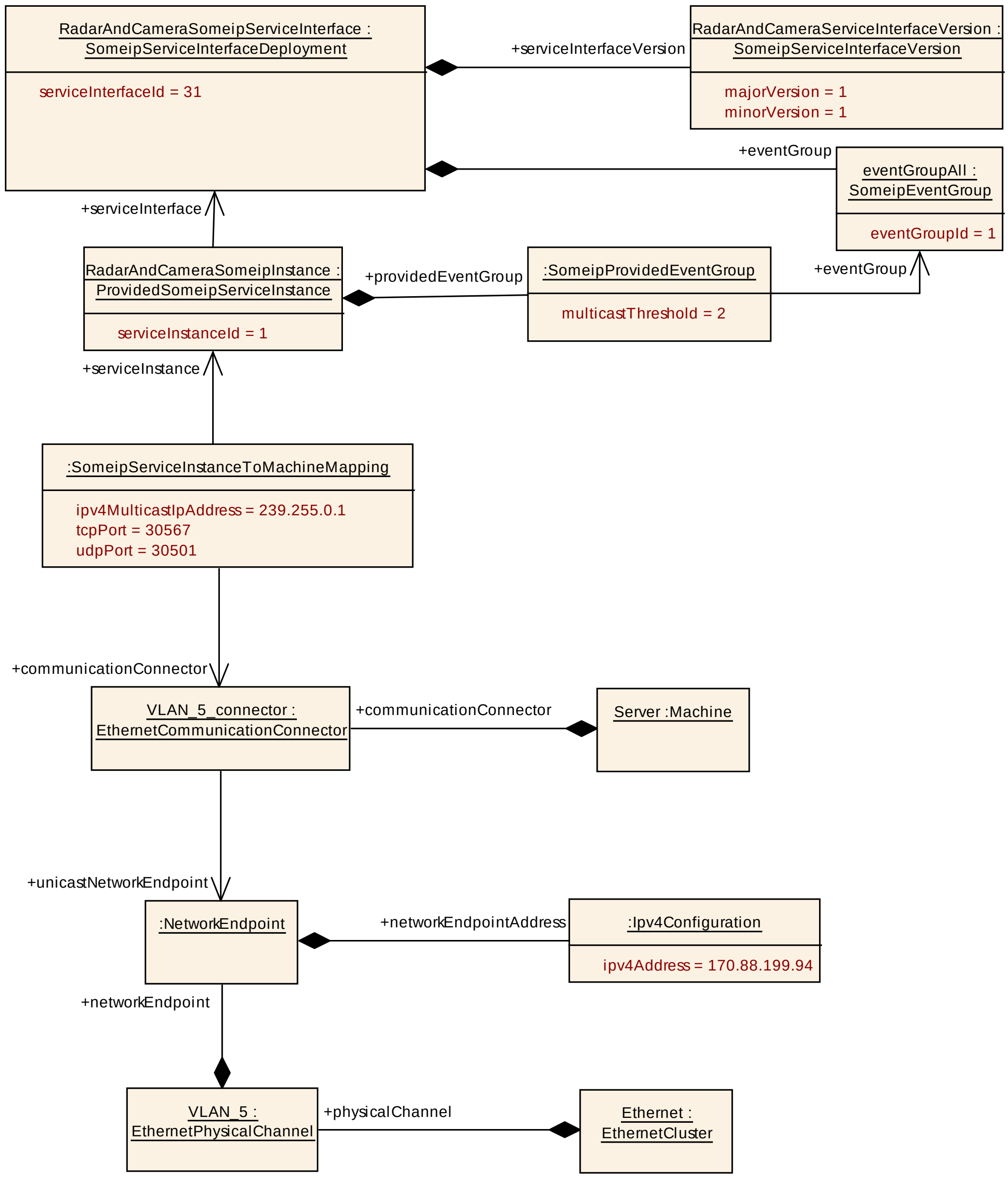

Figure A.12: SOME/IP Provided Service Instance

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), PortNumber (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.

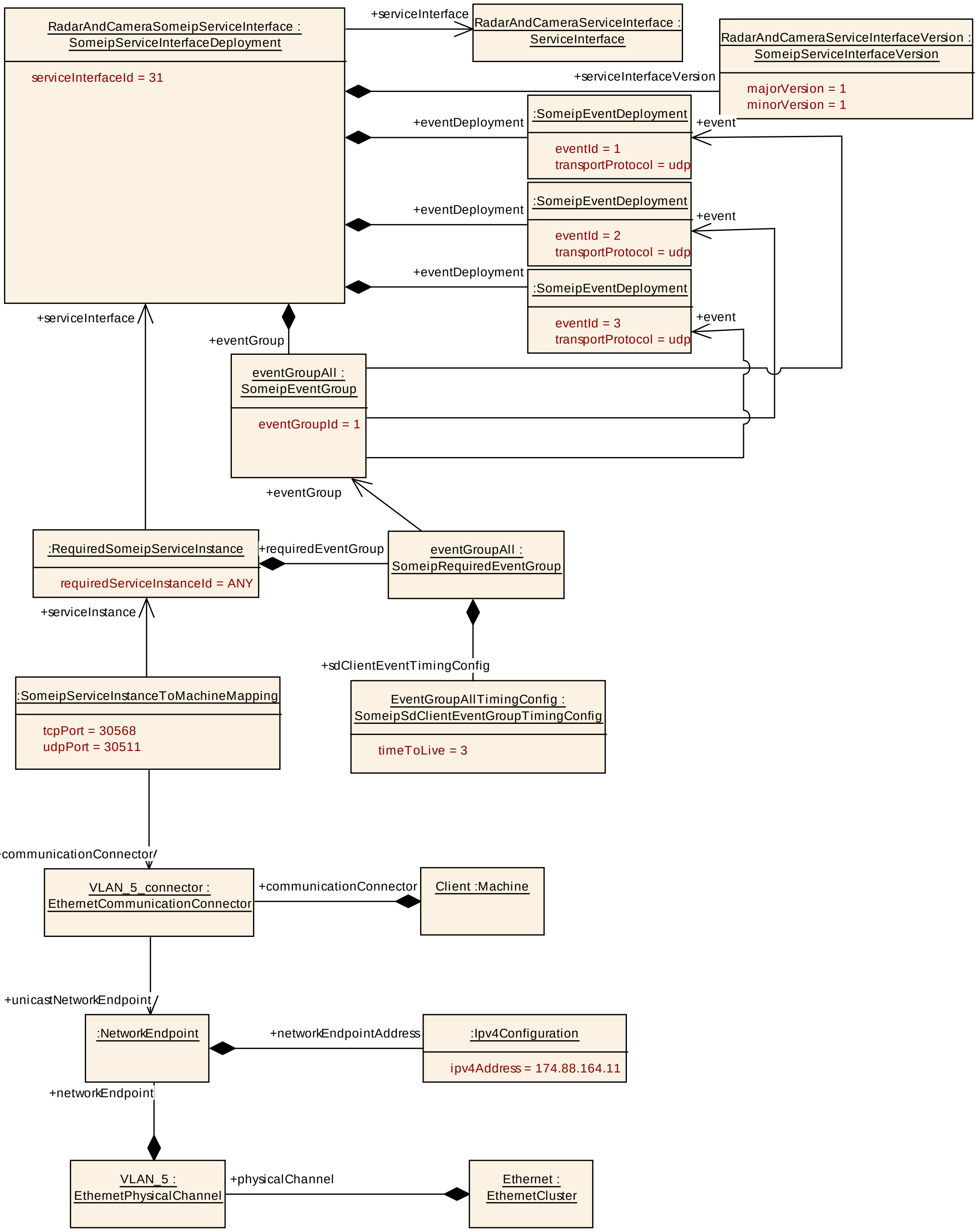

Figure A.13: SOME/IP Required Service Instance

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.sdClientEventTimingConfig.timeToLive = 3
- IPv4 Endpoint Option with IPv4 Address (170.88.164.11), Protocol (UDP), Port-Number (30511)

A.6 Signal-based communication example

The example in Figure A.14 sketches the modeling of a Signal-to-Service mapping.

In this example, the elements of the ServiceInterface *TestServiceInterface* that is referenced by the ProvidedSomeipServiceInstance are mapped to individual ISignalTriggerings. The *TestServiceInterface* contains only one single event *TestEvent* that is of type Structure and contains three members: x, y and z.

The ServiceInstanceToSignalMapping contains four SignalBasedEventElementToISignalTriggeringMappings. The *SignalBasedTestEvent* is mapped to an ISignalTriggering that is defined for an ISignalGroup. And the three eventElements are mapped to individual ISignalTriggerings for ISignals that will be transported in the enclosing ISignalGroup.

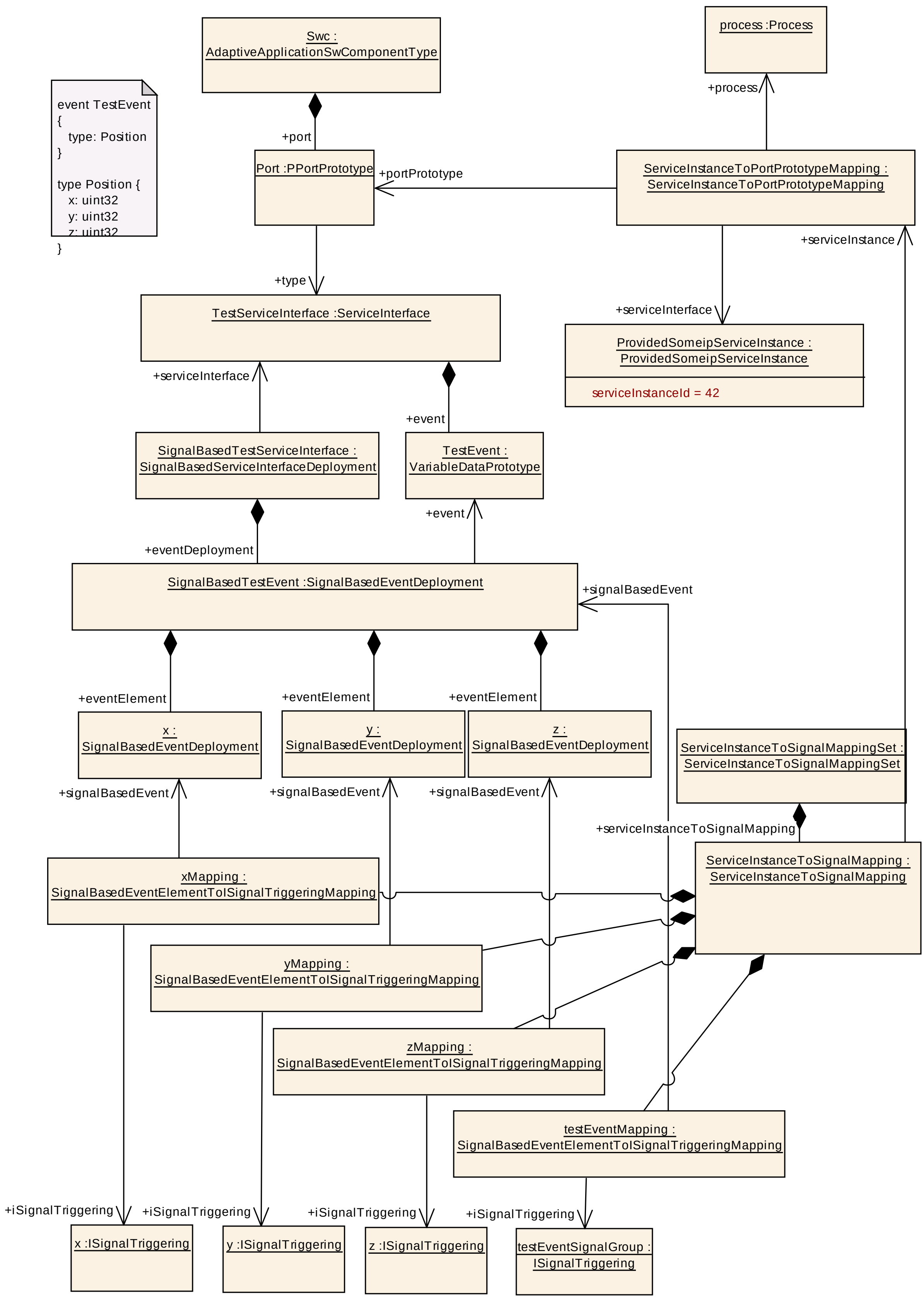

Figure A.14: Example for a Signal-to-Service mapping

A.7 Definition of Persistent Data

This chapter contains an example (see Figure A.15) for the modeling of persistent data storage starting form the design aspect down to the definition of the persistent storage and the mapping between design and deployment.

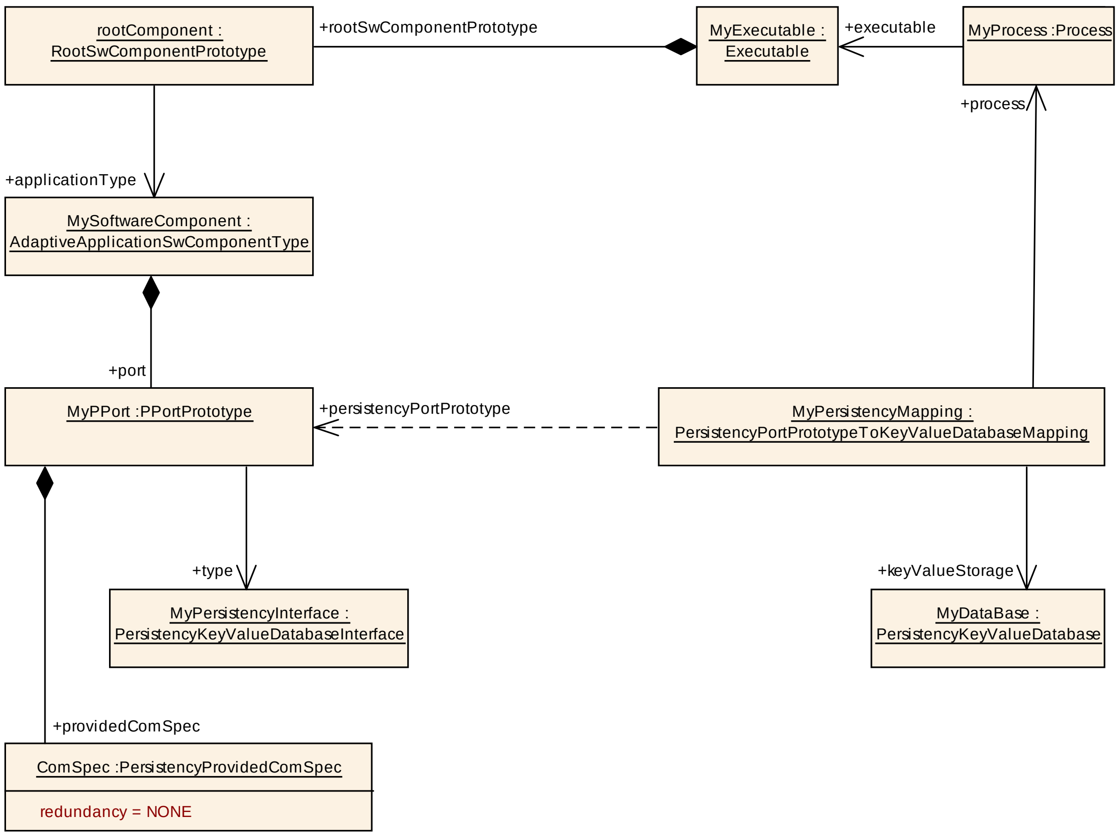

Figure A.15: Example modeling of persistent data (design + deployment)

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 CompositionSwComponentType

[TPS_MANI_01031] Semantics of CompositionDataPrototypeRef ⌈ The meta-class CompositionDataPrototypeRef has been created for the following purposes:

- Create a reference to a DataPrototype in the context of a CompositionSwComponentType. In this case it is not relevant whether the applicable sub-class of DataPrototype is typed by an ApplicationDataType or an ImplementationDataType. The aggregation CompositionDataPrototypeRef.dataPrototype shall be used.
- Create a reference to a DataPrototype located in a nested AutosarDataPrototype in the context of a CompositionSwComponentType. In this case it is technically relevant whether the applicable sub-class of DataPrototype is typed by an ApplicationDataType or an ImplementationDataType:
  - If the applicable sub-class of DataPrototype is typed by an ApplicationDataType then the aggregation in the role CompositionDataPrototypeRef.dataPrototype shall be used.
  - If the applicable sub-class of DataPrototype is typed by an ImplementationDataType then the aggregation in the role CompositionDataPrototypeRef.elementInImplDatatype **in addition** to CompositionDataPrototypeRef.dataPrototype shall be used.

⌋*()*

For referencing into the inside of a ImplementationDataType it is therefore necessary to use the aggregation CompositionDataPrototypeRef.dataPrototype to “get to” the root DataPrototype and **then proceed into the guts of the composite ImplementationDataType** by means of using CompositionDataPrototypeRef.elementInImplDatatype.

[constr_1480] Mutual existence of CompositionDataPrototypeRef.elementInImplDatatype vs. attributes of CompositionDataPrototypeRef.dataPrototype ⌈ If the aggregation CompositionDataPrototypeRef.elementInImplDatatype exists then the following attributes shall not exist:

- CompositionDataPrototypeRef.dataPrototype.rootDataPrototype

- CompositionDataPrototypeRef.dataPrototype.contextDataPrototype

⌋()

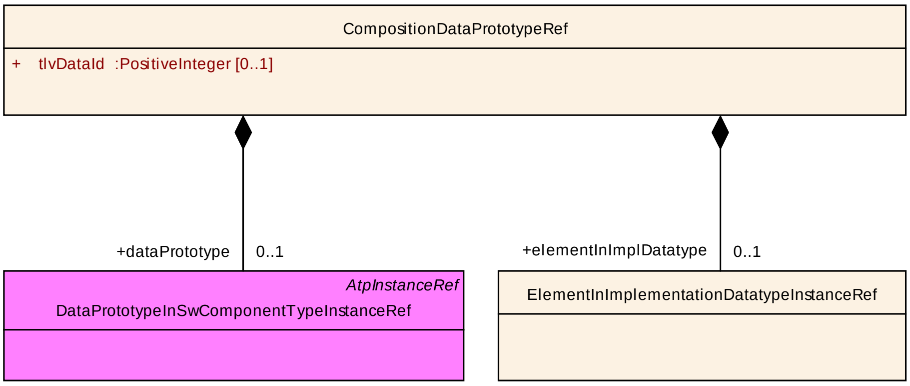

Figure B.1: Modeling of CompositionDataPrototypeRef

[constr_1481] Usage of CompositionDataPrototypeRef in the *AUTOSAR adaptive platform* ⌈ If CompositionDataPrototypeRef is used in the context of the *AUTOSAR adaptive platform* then the actual DataPrototypeInSwComponentTypeInstanceRef.targetDataPrototype shall be either a VariableDataPrototype or an ArgumentDataPrototype. ⌋()

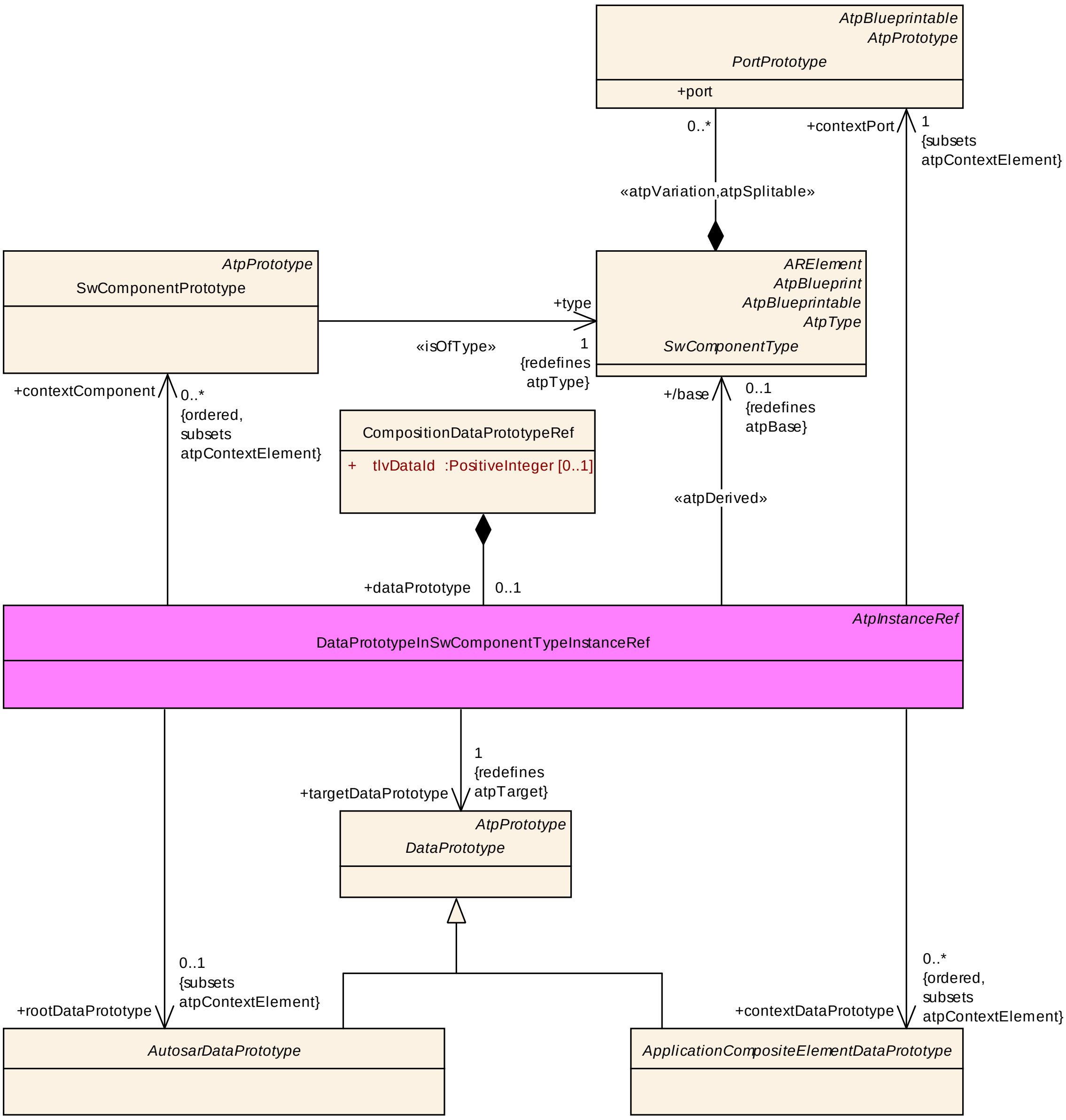

Figure B.2: Modeling of DataPrototypeInSwComponentTypeInstanceRef

Class	**CompositionDataPrototypeRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to refer to an AUTOSAR DataPrototype in the context of a CompositionSwComponentType. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataPrototype	DataPrototype	0..1	iref	This attribute shall exist if the InstanceRef points to a DataPrototype typed by an ApplicationDataType. **Tags:** atp.Status=draft

elementInImplDatatype	ElementInImplementationDatatypeInstanceRef	0..1	aggr	This attribute shall exist if the InstanceRef points to a DataPrototype typed by an ImplementationDataType. **Tags:** atp.Status=draft
tlvDataId	PositiveInteger	0..1	attr	This attribute represents the ability to specify a TLV data-id for the serialization of a specific DataPrototype in the context of a (potentially deeply-nested) composite data structure for the case that the data structure has optional elements. This value does not represent the entire value of the tag, e.g. the wire-type is not included (because it can be derived from the information about the underlying AutosarDataType).

Table B.1: CompositionDataPrototypeRef

Class	**DataPrototypeInSwComponentTypeInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to: • refer to a DataPrototype in the context of a CompositionSwComponentType. • refer to the internal structure of a DataPrototype in the context of a CompositionSwComponentType. **Tags:** atp.Status=draft			
Base	ARObject,AtpInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	SwComponentType	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft xml.sequenceOffset=10
contextComponent (ordered)	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
contextDataPrototype (ordered)	ApplicationCompositeElementDataPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=50
contextPort	PortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
rootDataPrototype	AutosarDataPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40
targetDataPrototype	DataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=60

Table B.2: DataPrototypeInSwComponentTypeInstanceRef

Class	ArVariableInImplementationDataInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Data Elements			
Note	This class represents the ability to navigate into a data element inside of an VariableDataPrototype which is typed by an ImplementationDatatype. Note that it shall not be used if the target is the VariableDataPrototype itself (e.g. if its a primitive). Note that this class follows the pattern of an InstanceRef but is not implemented based on the abstract classes because the ImplementationDataType isn't either, especially because ImplementationDataTypeElement isn't derived from AtpPrototype.			
Base	ARObject			
Attribute	Type	Mul.	Kind	Note
contextDataPrototype (ordered)	ImplementationDataTypeElement	*	ref	This is a context in case there are subelements with explicit types. The reference has to be ordered to properly reflect the nested structure. **Tags:** xml.sequenceOffset=30
portPrototype	PortPrototype	0..1	ref	This is the port providing/receiving the root of the variable **Tags:** xml.sequenceOffset=10
rootVariableDataPrototype	VariableDataPrototype	0..1	ref	This refers to the variableDataPrototype which is typed by the implementationDatatype in which which the target can be found. **Tags:** xml.sequenceOffset=20
targetDataPrototype	ImplementationDataTypeElement	1	ref	This is a context in case there are subelements with explicit types. **Tags:** xml.sequenceOffset=40

Table B.3: ArVariableInImplementationDataInstanceRef

B.2 Modeling of InstanceRefs

This section illustrates the concrete modeling of the instance references used in the previous parts of this document.

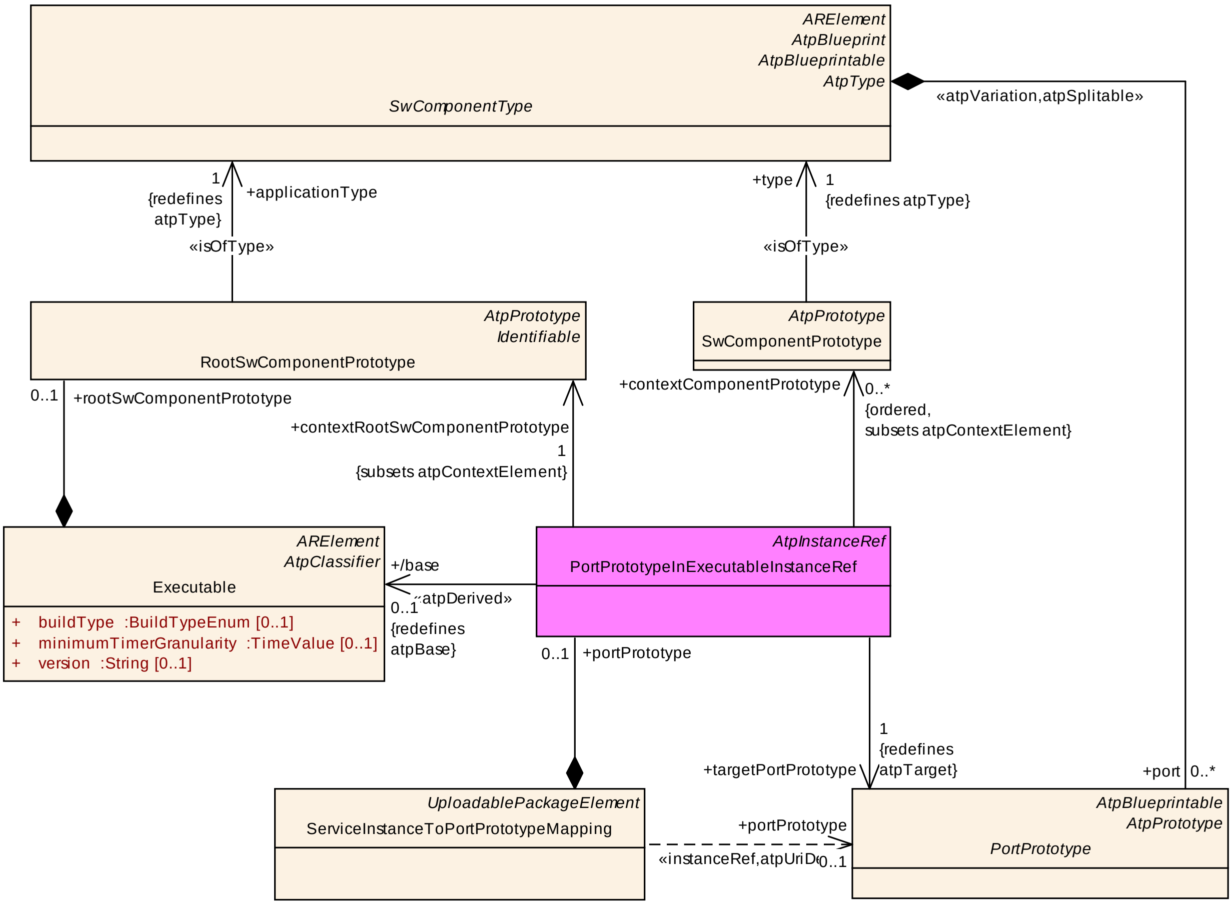

Figure B.3: Modeling of PortPrototypeInExecutableInstanceRef

Class	**PortPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process::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
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.4: PortPrototypeInExecutableInstanceRef

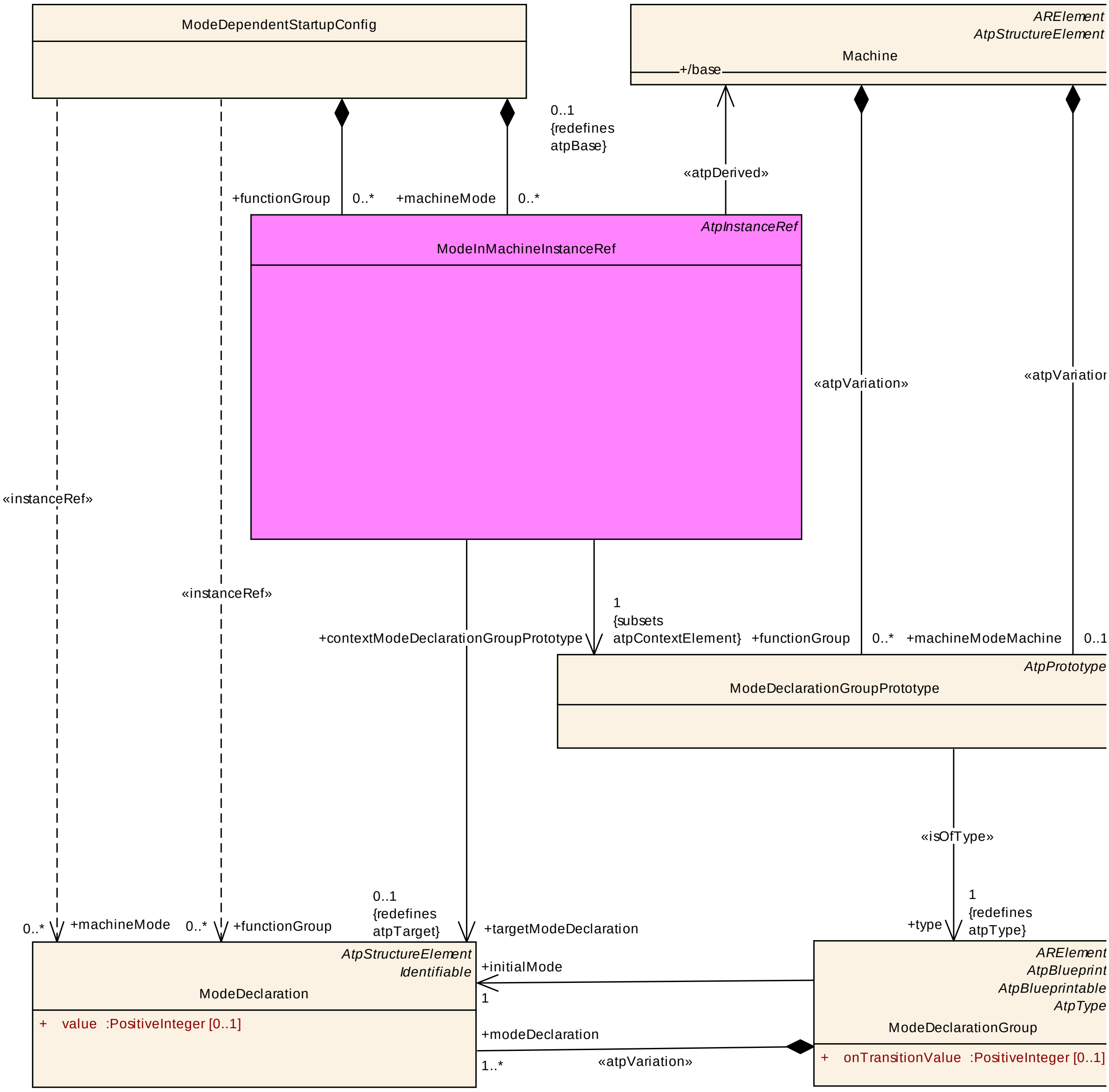

Figure B.4: Modeling of ModeInMachineInstanceRef

Class	**ModeInMachineInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process::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

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.5: ModeInMachineInstanceRef

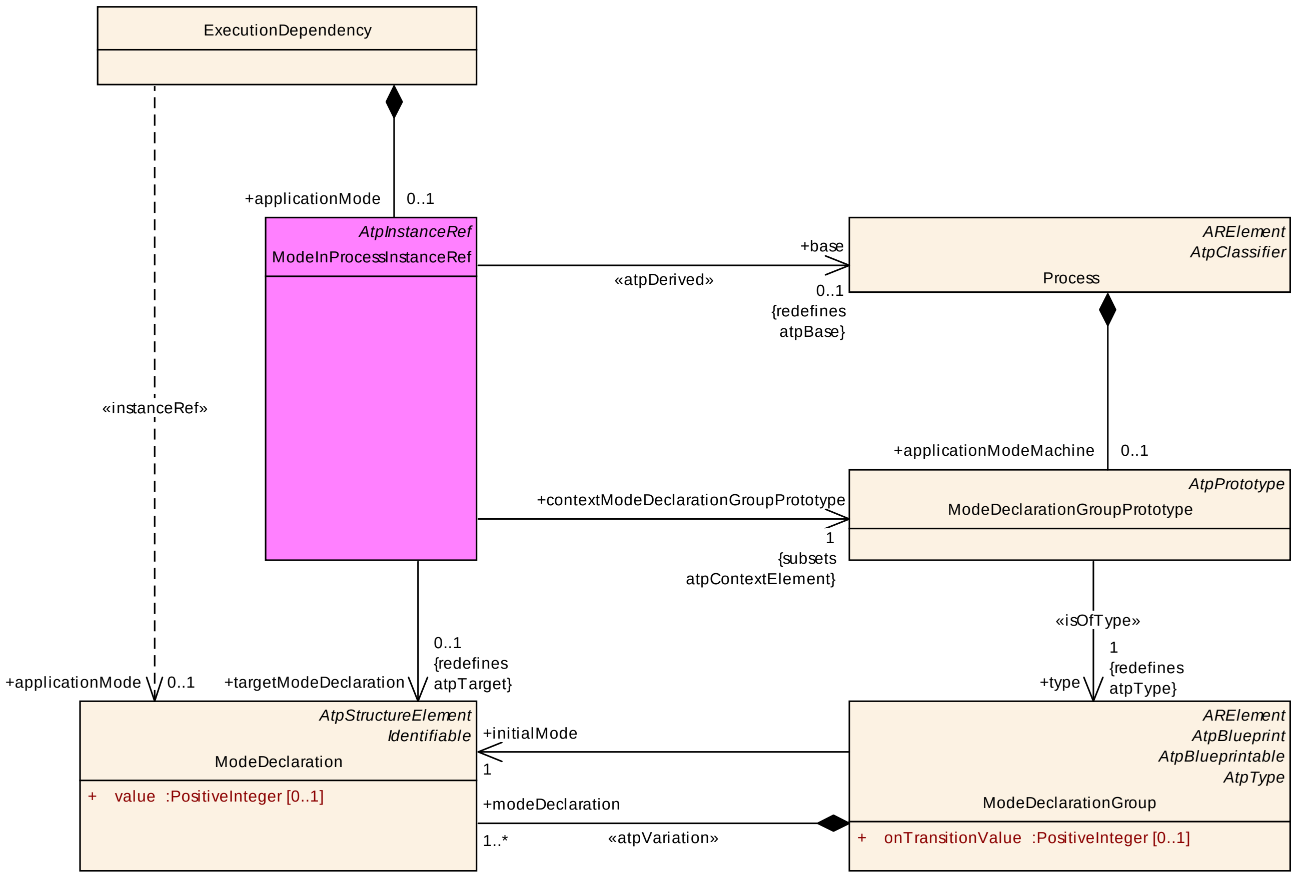

Figure B.5: Modeling of ModeInProcessInstanceRef

Class	**ModeInProcessInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process::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.6: ModeInProcessInstanceRef

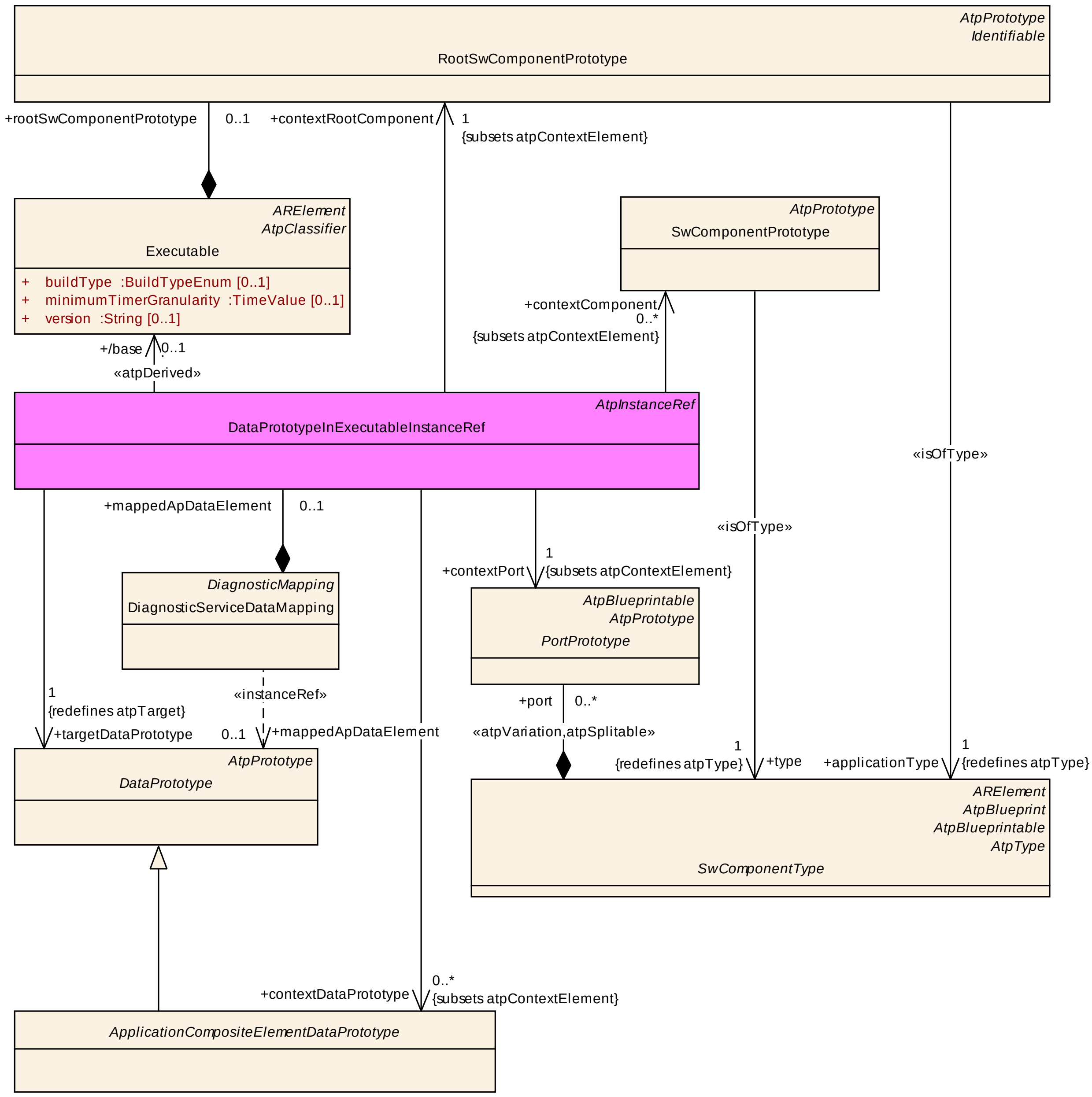

Figure B.6: Modeling of `DiagnosticServiceDataMapping` via `DataPrototypeInExecutableInstanceRef`

Class	**DataPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::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
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.7: DataPrototypeInExecutableInstanceRef

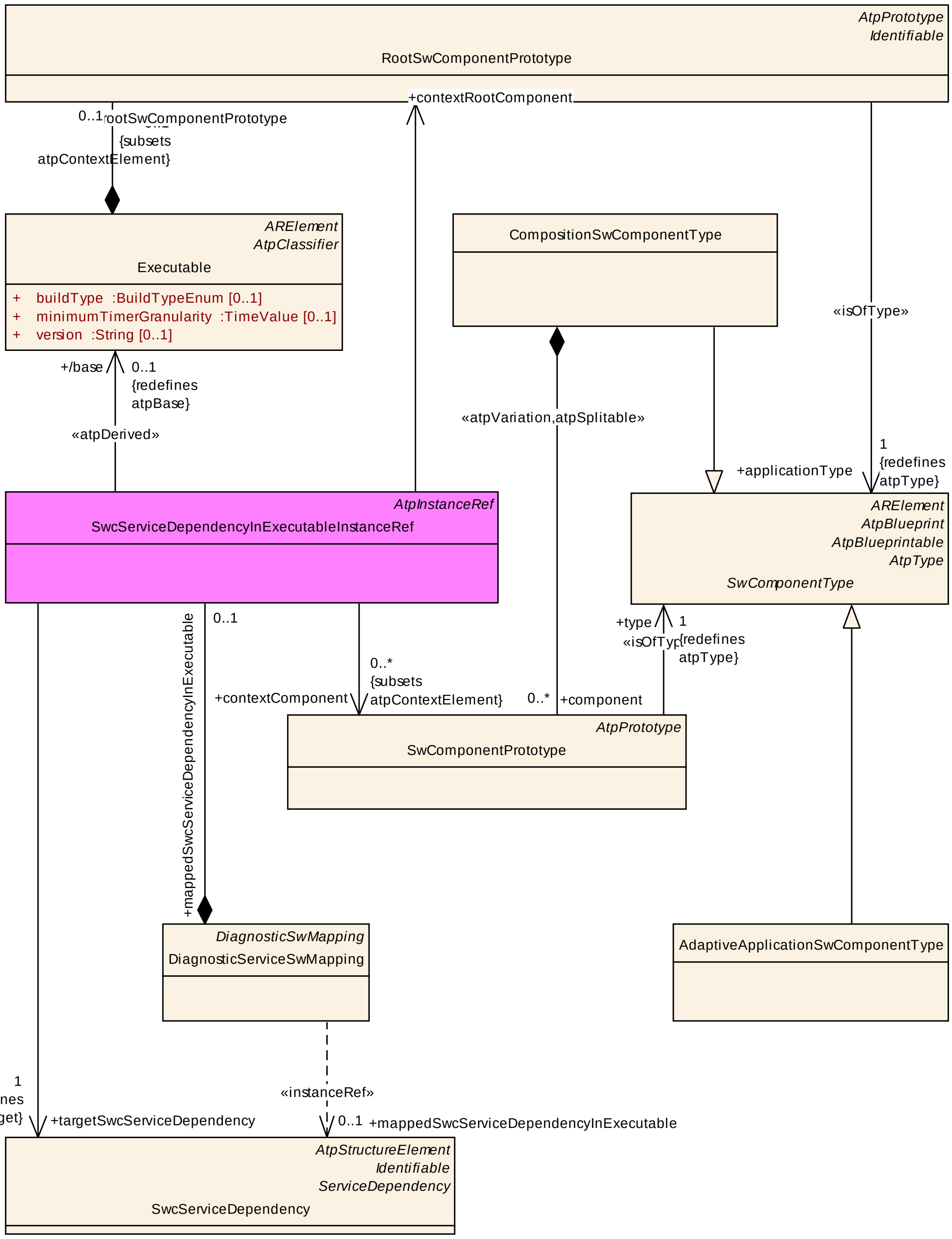

Figure B.7: Modeling of DiagnosticServiceSwMapping via SwcServiceDependencyInExecutableInstanceRef

Class	**SwcServiceDependencyInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::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.8: SwcServiceDependencyInExecutableInstanceRef

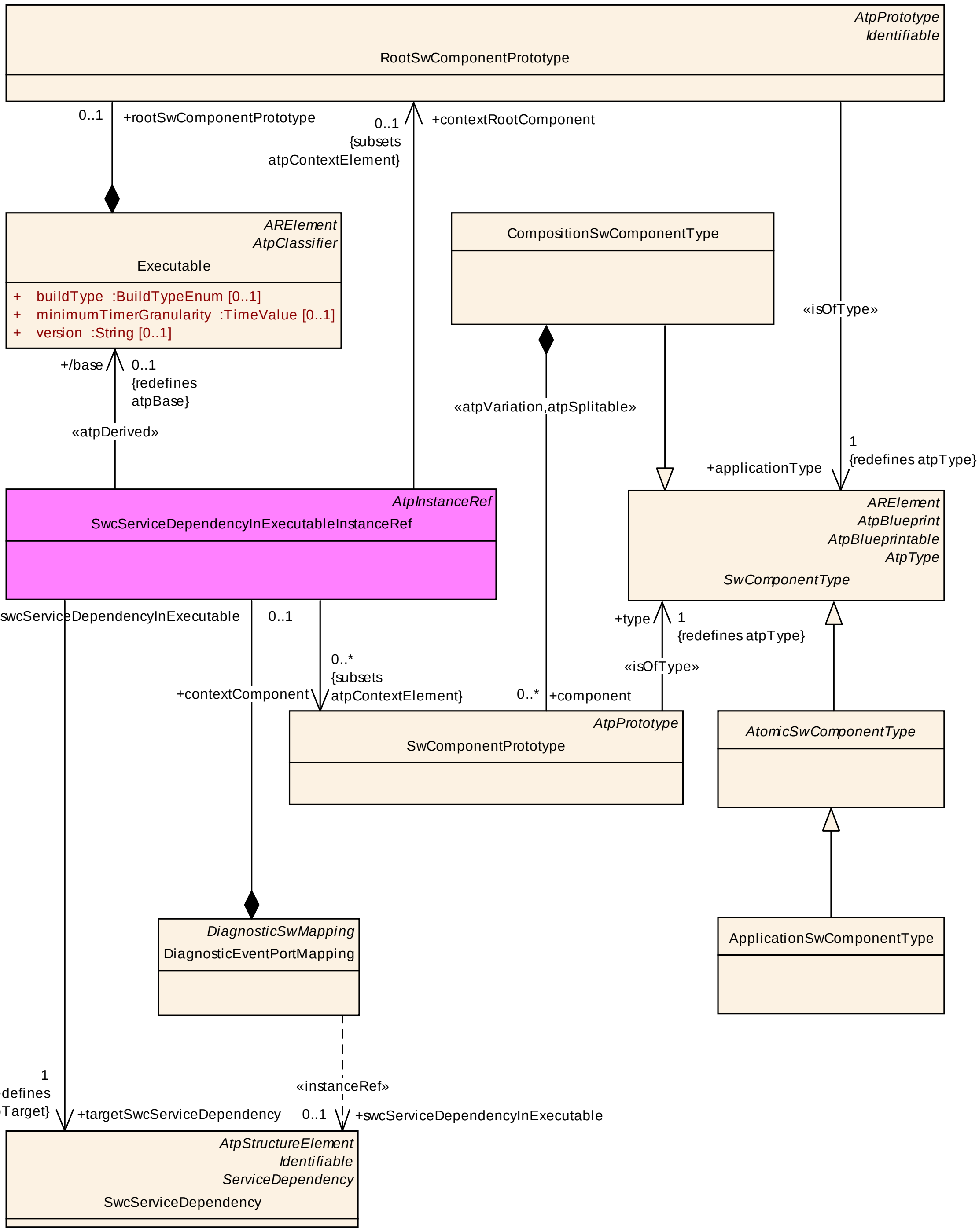

Figure B.8: Modeling of `DiagnosticEventPortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

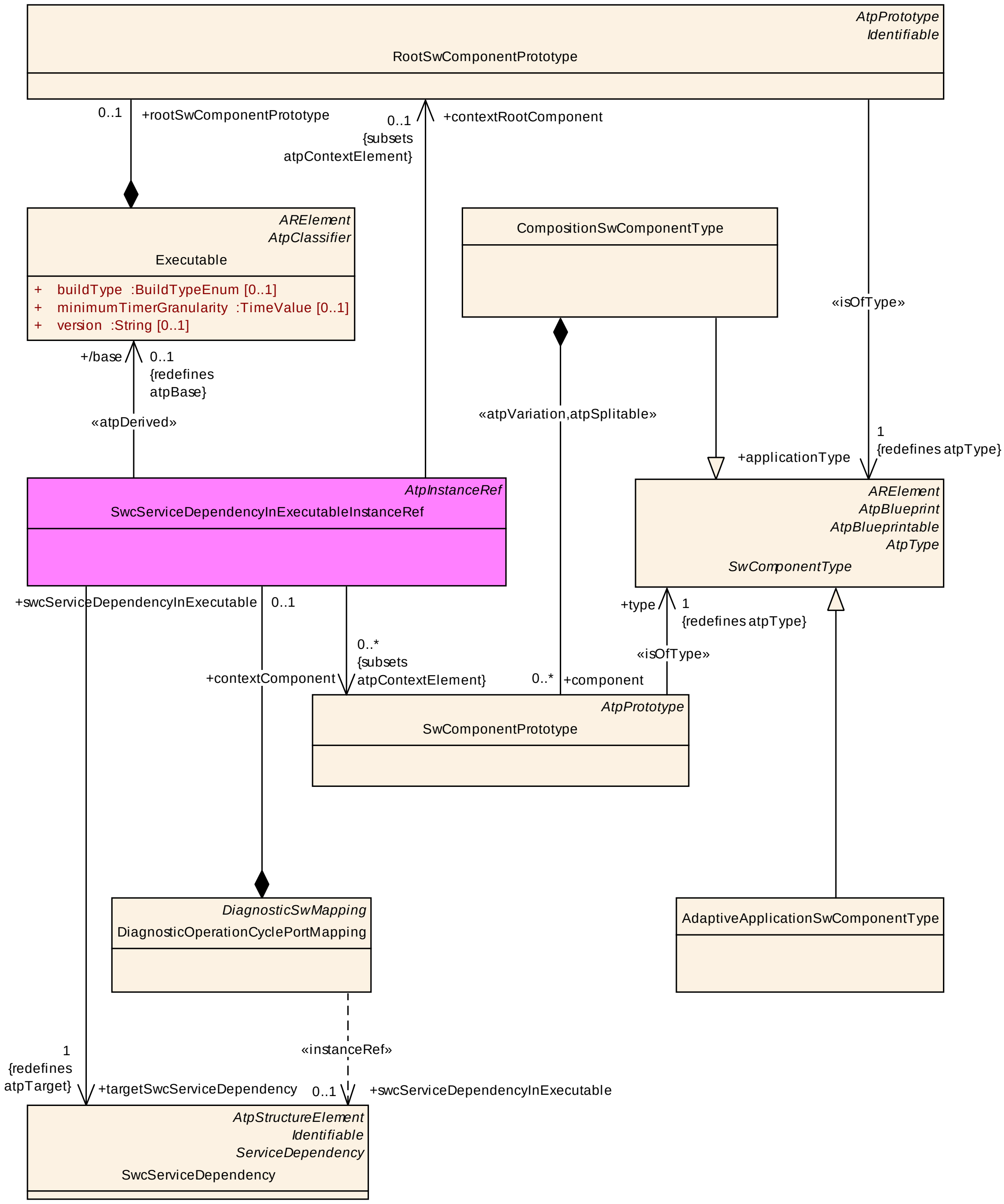

Figure B.9: Modeling of DiagnosticOperationCyclePortMapping via SwcServiceDependencyInExecutableInstanceRef

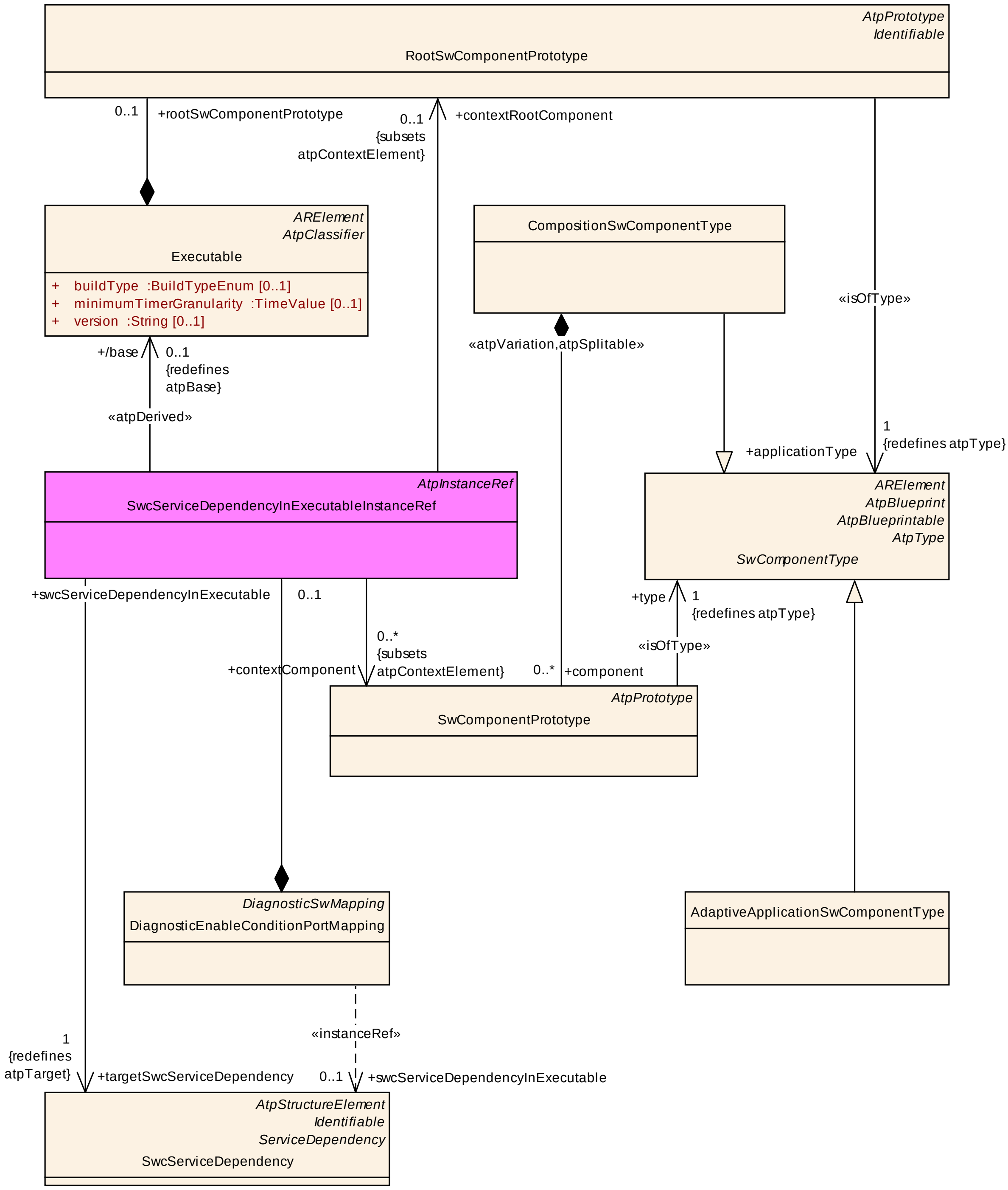

Figure B.10: Modeling of `DiagnosticEnableConditionPortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

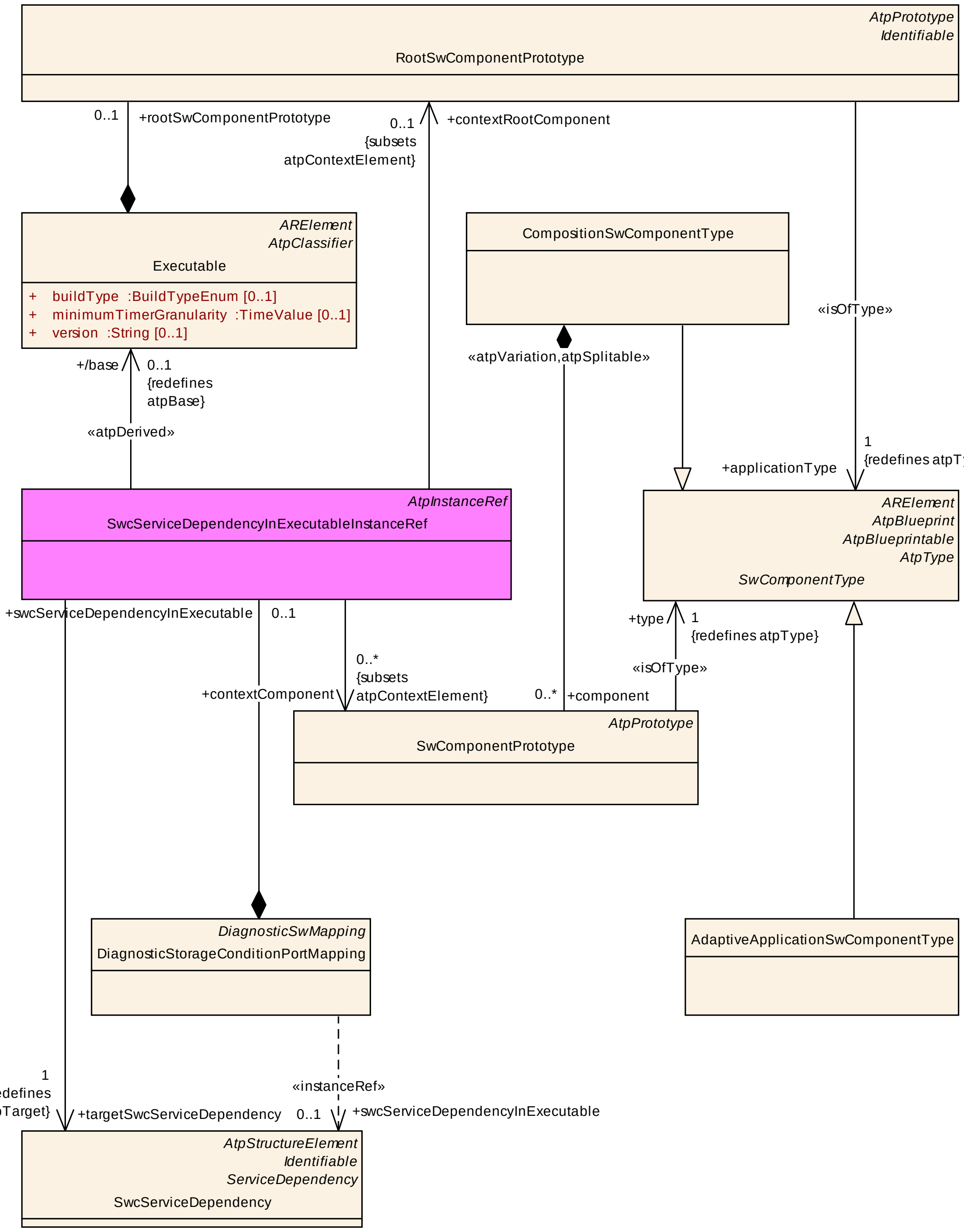

Figure B.11: Modeling of DiagnosticStorageConditionPortMapping via SwcServiceDependencyInExecutableInstanceRef

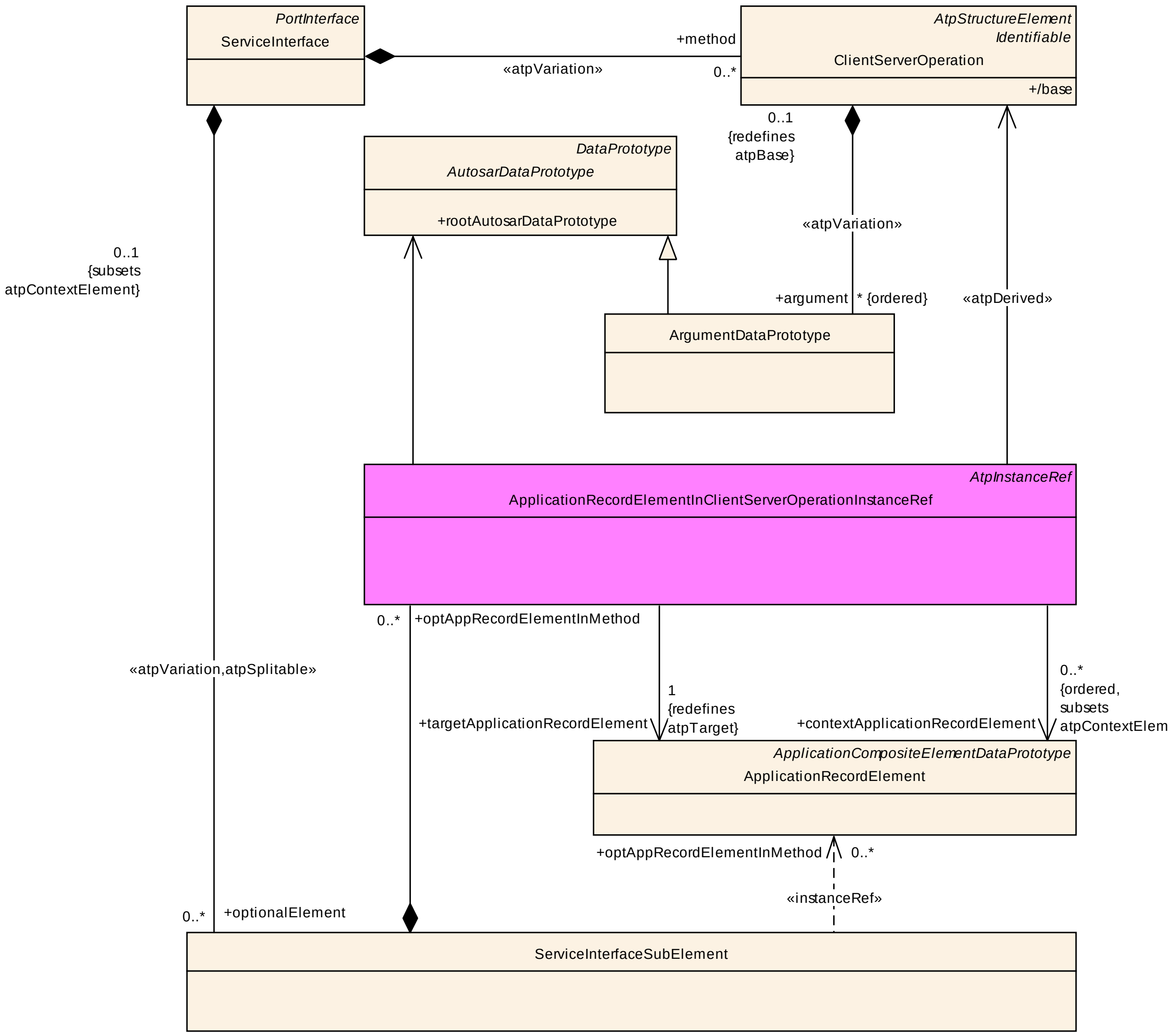

Figure B.12: Modeling of ApplicationRecordElementInClientServerOperationInstanceRef

Class	**ApplicationRecordElementInClientServerOperationInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::OptionalElementInServiceInterface			
Note	**Tags:** atp.Status=draft			
Base	ARObject,AtpInstanceRef			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
base	ClientServerOperation	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft
contextApplicationRecordElement (ordered)	ApplicationRecordElement	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20

rootAutosarDataPrototype	AutosarDataPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetApplicationRecordElement	ApplicationRecordElement	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.9: ApplicationRecordElementInClientServerOperationInstanceRef

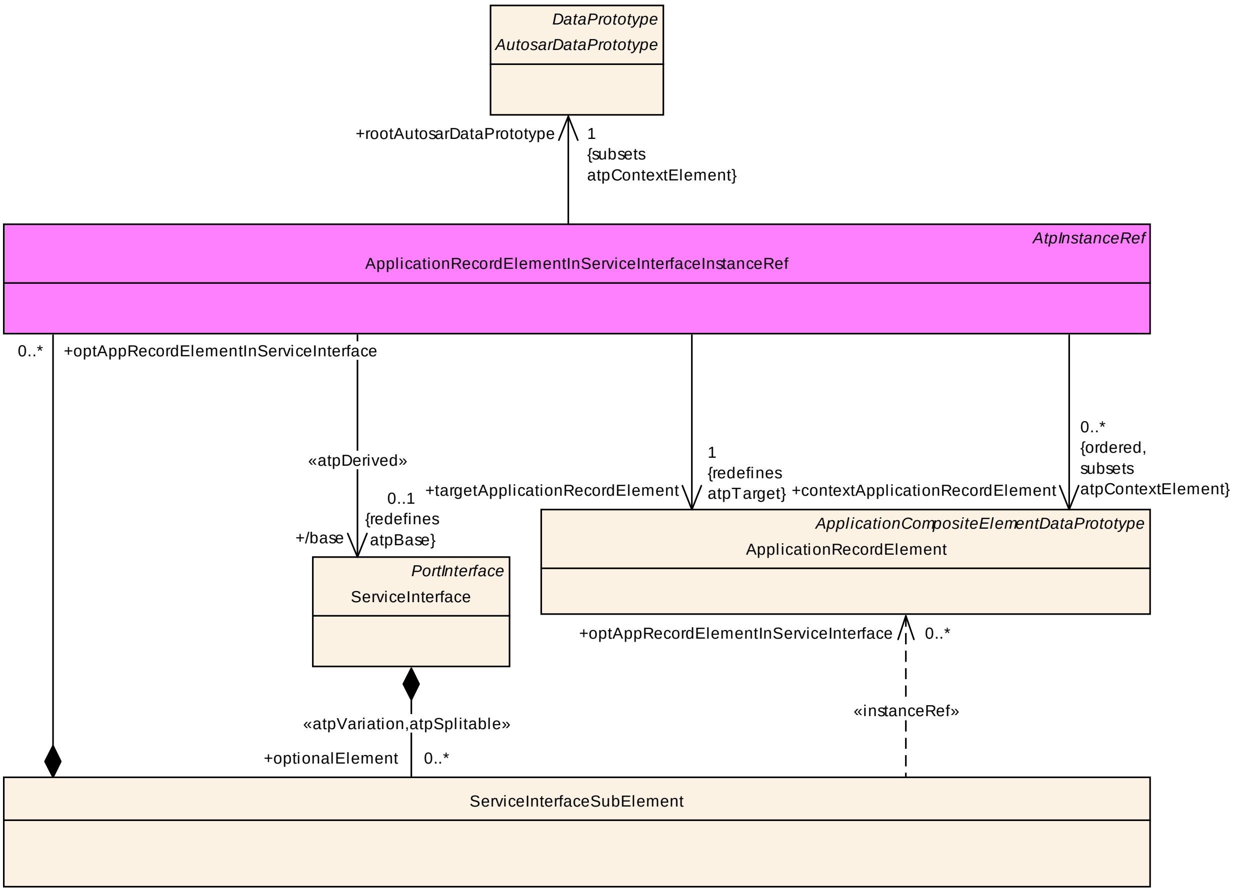

Figure B.13: Modeling of ApplicationRecordElementInServiceInterfaceInstanceRef

Class	**ApplicationRecordElementInServiceInterfaceInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::OptionalElementInServiceInterface			
Note	This meta-class represents the ability to establish an InstanceRef to an ApplicationRecordElement in the context of an AutosarDataPrototype that is directly or indirectly owned by a ServiceInterface. **Tags:** atp.Status=draft			
Base	ARObject,AtpInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	ServiceInterface	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft

contextApplicationRecordElement (ordered)	ApplicationRecordElement	*	ref	This represents the collection of context ApplicationRecordElements. **Tags:** atp.Status=draft xml.sequenceOffset=30
rootAutosarDataPrototype	AutosarDataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetApplicationRecordElement	ApplicationRecordElement	1	ref	This reference points to the target record element. **Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.10: ApplicationRecordElementInServiceInterfaceInstanceRef

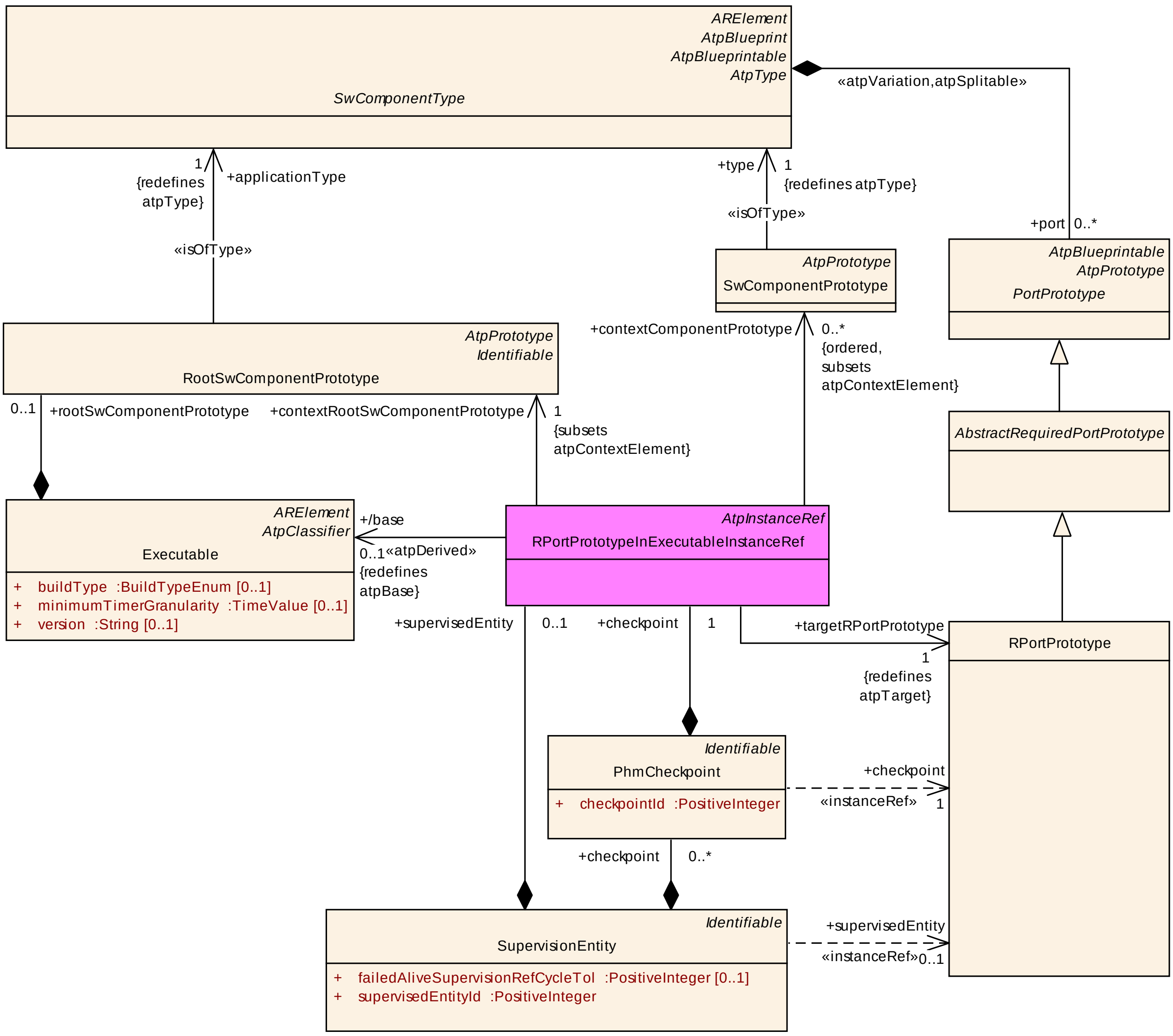

Figure B.14: Modeling of RPortPrototypeInExecutableInstanceRef

Class	**RPortPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::PlatformHealthManagement::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
contextComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSwComponentPrototype	RootSwComponentPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetRPortPrototype	RPortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.11: RPortPrototypeInExecutableInstanceRef

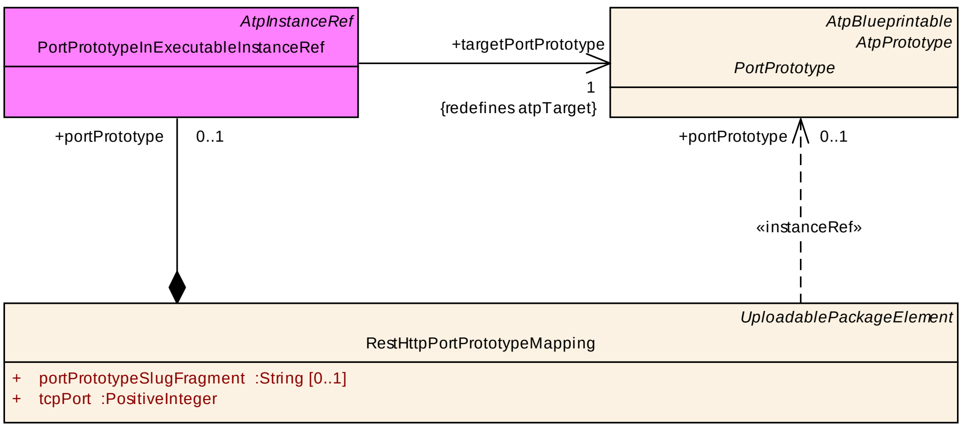

Figure B.15: Modeling of reference `RestHttpPortPrototypeMapping.portPrototype`

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, Packageable Element, Referrable			
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*
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, variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableEle ment	*	aggr	Elements that are part of this package **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20

referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortLabel xml.sequenceOffset=10

Table C.2: ARPackage

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 MultilanguagePlainText. The content of each entry can be used for illustration of the language. The used language itself depends on the language attribute in the entry. **Tags:** xml.sequenceOffset=30

Table C.3: AdminData

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 C.4: ApplicationRecordDataType

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

Table C.5: ApplicationSwComponentType

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	1..*	aggr	The value for a single array element. All ValueSpecifications aggregated by ArrayValueSpecification shall have the same structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table C.6: 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.7: 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			
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.8: AtomicSwComponentType

Class	AtpInstanceRef (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::AbstractStructure			
Note	An M0 instance of a classifier may be represented as a tree rooted at that instance, where under each node come the sub-trees representing the instances which act as features under that node. An instance ref specifies a navigation path from any M0 tree-instance of the base (which is a classifier) to a leaf (which is an instance of the target).			
Base	ARObject			
Attribute	Type	Mul.	Kind	Note
atpBase	AtpClassifier	1	ref	This is the base from which the navigaion path starts. **Stereotypes:** atpAbstract; atpDerived
atpContextElement (ordered)	AtpPrototype	*	ref	This is one particular step in the navigation path. **Stereotypes:** atpAbstract
atpTarget	AtpFeature	1	ref	This is the target of the instance ref. In other words it is the terminal of the navigation path. **Stereotypes:** atpAbstract

Table C.9: AtpInstanceRef

Class	**AutosarDataPrototype (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of an AutosarDataType.			
Base	ARObject, AtpFeature, AtpPrototype, DataPrototype, Identifiable, Multilanguage Referrable, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
type	AutosarDataType	1	tref	This represents the corresponding data type. **Stereotypes:** isOfType

Table C.10: 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			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType.

Table C.11: AutosarDataType

Class	**BaseType (abstract)**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This abstract meta-class represents the ability to specify a platform dependant base type.			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
baseTypeDefinition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Table C.12: BaseType

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**

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.13: ClientServerInterface

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, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:** xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:** xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:** xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:** xml.sequenceOffset=30

Table C.14: CompuMethod

Class	**DataPrototype (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of any data type.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

swDataDefProps	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level.

Table C.15: DataPrototype

Class	**DiagnosticContributionSet**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticContribution			
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 DiagnosticContributionSet. **Tags:** atp.recommendedPackage=DiagnosticContributionSets			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
commonProperties	DiagnosticCommonProps	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire DiagnosticContributionSet. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=commonProperties
element	DiagnosticCommonElement	*	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 DiagnosticContributionSet. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceTable, variationPoint.shortLabel vh.latestBindingTime=postBuild

Table C.16: DiagnosticContributionSet

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			
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.17: DiagnosticServiceInstance

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.			
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
soAdConfig	SoAdConfig	0..1	aggr	SoAd Configuration for one specific Physical Channel.
vlan	VlanConfig	0..1	aggr	VLAN Configuration.

Table C.18: EthernetPhysicalChannel

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, PackageableElement, Referrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataTransformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataTransformation, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
dataTypePolicy	DataTypePolicyEnum	1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=iSignalProps

iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in an 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 "InitValue". If a full DataMapping exist for the SystemSignal this information may be available from a configured SenderComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.
length	Integer	1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.
networkRepresentationProps	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. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalidValue" and the Data Semantics.
systemSignal	SystemSignal	1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal.

timeoutSubstitutionValue	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal.
transformationISignalProps	TransformationISignalProps	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class.

Table C.19: ISignal

Class	**ISignalGroup**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	SignalGroup of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal Group is sent in different SignalIPdus to multiple receivers. An ISignalGroup refers to a set of ISignals that shall always be kept together. A ISignalGroup represents a COM Signal Group. Therefore it is recommended to put the ISignalGroup in the same Package as ISignals (see atp.recommendedPackage) **Tags:** atp.recommendedPackage=ISignalGroup			
Base	ARObject, CollectableElement, FibexElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
comBasedSignalGroupTransformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignalGroup based on the COMBasedTransformer approach. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=comBasedSignalGroupTransformation, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
iSignal	ISignal	*	ref	Reference to a set of ISignals that shall always be kept together.
systemSignalGroup	SystemSignalGroup	1	ref	Reference to the SystemSignalGroup that is defined on VFB level and that is supposed to be transmitted in the ISignalGroup.
transformationISignalProps	TransformationISignalProps	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignalGroup specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignalGroups are described in the TransformationTechnology class.

Table C.20: ISignalGroup

Class	**ISignalIPdu**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. **Tags:** atp.recommendedPackage=Pdus			
Base	ARObject, CollectableElement, FibexElement, IPdu, Identifiable, Multilanguage Referrable, PackageableElement, Pdu, Referrable			
Attribute	**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 SignalIPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
pduCounter	SignalIPduCounter	0..1	aggr	An included Pdu counter is used to ensure that a sequence of Pdus is maintained. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
pduReplication	SignalIPduReplication	0..1	aggr	Pdu Replication is a form of redundancy where the data content of one ISignalIPdu (source) is transmitted inside a set of replica ISignalIPdus. These ISignalIPdus (copies) have different Pdu IDs, identical PduCounters, identical data content and are transmitted with the same frequency. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unusedBitPattern	Integer	1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table C.21: 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 ISignalTriggering-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.22: ISignalTriggering

Class	Identifiable (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
Base	ARObject, MultilanguageReferrable, Referrable			
Attribute	Type	Mul.	Kind	Note
desc	MultiLanguageOverviewParagraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** xml.sequenceOffset=-60
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:** xml.sequenceOffset=-40

annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
introduction	Documentation Block	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.23: Identifiable

Class	**NonOsModuleInstantiation (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::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.Status=draft			
Base	ARObject, AdaptiveModuleInstantiation, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.24: 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			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table C.25: 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
providedInterface	PortInterface	1	tref	The interface that this port provides. **Stereotypes:** isOfType

Table C.26: PPortPrototype

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

Table C.27: PRPortPrototype

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, PackageableElement, Referrable			
Attribute	Type	Mul.	Kind	Note

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.28: Pdu

Class	**PersistencyPortPrototypeToKeyValueDatabaseMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a PortPrototype and a key used in a persistent storage. **Tags:** atp.Status=draft; atp.recommendedPackage=PersistentPortPrototypeToKeyValueDatabaseMappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadablePackageElement			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
keyValueStorage	PersistencyKeyValueDatabase	0..1	ref	This reference represents the mapped key-value storage. **Tags:** atp.Status=draft
persistencyPortPrototype	PortPrototype	0..1	iref	This reference represents the affected Persistency PortPrototype **Tags:** atp.Status=draft
process	Process	0..1	ref	This reference represents the process required for context of the mapping. **Tags:** atp.Status=draft

Table C.29: PersistencyPortPrototypeToKeyValueDatabaseMapping

Class	**PersistencyProvidedComSpec**			
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***
redundancy	PersistencyRedundancyEnum	0..1	attr	This attribute represents a requirement towards the redundancy of storage.

Table C.30: PersistencyProvidedComSpec

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, PackageableElement, 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
luminousIntensityExp	Numerical	0..1	attr	The exponent of the physical dimension "luminous intensity". **Tags:** xml.sequenceOffset=80
massExp	Numerical	0..1	attr	The exponent of the physical dimension "mass". **Tags:** xml.sequenceOffset=30
molarAmountExp	Numerical	0..1	attr	The exponent of the physical dimension "quantity of substance". **Tags:** xml.sequenceOffset=70
temperatureExp	Numerical	0..1	attr	The exponent of the physical dimension "temperature". **Tags:** xml.sequenceOffset=60
timeExp	Numerical	0..1	attr	The exponent of the physical dimension "time". **Tags:** xml.sequenceOffset=40

Table C.31: PhysicalDimension

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=ServiceInterfaceToDataType Mappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Type	Mul.	Kind	Note
dataTypeMappingSet	DataTypeMappingSet	1..*	ref	This represents the reference to the applicable dataTypemappingSet **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform
portInterface	PortInterface	1	ref	This represents the reference to the applicable PortInterface **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform

Table C.32: PortInterfaceToDataTypeMapping

Class	PortPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable, Multilanguage Referrable, Referrable			
Attribute	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.33: 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.
instanceIdentifier	PositiveInteger	0..1	attr	Instance identifier. Can be used for e.g. service discovery to identify the instance of the service.
priority	PositiveInteger	0..1	attr	Priority defined per provided ServiceInstance.
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.34: ProvidedServiceInstance

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

Table C.35: 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**
requiredInterface	PortInterface	1	tref	The interface that this port requires, i.e. the port depends on another port providing the specified interface. **Stereotypes:** isOfType

Table C.36: 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 ValueSpecification indepenently of the shortNames. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table C.37: RecordValueSpecification

Class	**Referrable (abstract)**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	ARObject			
Attribute	**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.38: Referrable

Class	**RoleBasedPortAssignment**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
Note	This class specifies an assignment of a role to a particular service port (RPortPrototype or PPortPrototype) of an AtomicSwComponentType. With this assignment, the role of the service port can be mapped to a specific ServiceNeeds element, so that a tool is able to create the correct connector.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
portPrototype	PortPrototype	1	ref	Service PortPrototype used in the assigned role. This PortPrototype shall either belong to the same AtomicSwComponentType as the SwcInternalBehavior which owns the ServiceDependency or to the same NvBlockSwComponentType 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 PortInterface as standardized in the Software Specification of the related AUTOSAR Service.

Table C.39: 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

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.40: 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 sdgCaption 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
sdgContentsType	SdgContents	0..1	aggr	This is the content of the Sdg. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

Table C.41: Sdg

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.
invalidationPolicy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement

Table C.42: 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. The queue is implemented by the RTE. The value shall be greater or equal to 1. Setting the value of queueLength to 1 implies that incoming requests are rejected while another request that arrived earlier is being processed.
transformationComSpecProps	TransformationComSpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table C.43: 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			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table C.44: 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, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table C.45: ServiceSwComponentType

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.46: 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			
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.47: 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 DataTypes in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddrMethod, swPointerTagetProps, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue • Code generation policy provided by swRecordLayout **Tags:** vh.latestBindingTime=codeGenerationTime			
Base	ARObject			
Attribute	**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
implementationDataType	ImplementationDataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTargetProps), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:** xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:** xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:** xml.sequenceOffset=30

swAlignment	AlignmentType	0..1	attr	The attribute describes the intended alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced SwAddrMethod. **Tags:** xml.sequenceOffset=33
swBitRepresentation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:** xml.sequenceOffset=60
swCalibrationAccess	SwCalibrationAccessEnum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:** xml.sequenceOffset=70
swCalprmAxisSet	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:** xml.sequenceOffset=90
swComparisonVariable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170; xml.typeElement=false
swDataDependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:** xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220; xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:** xml.sequenceOffset=230

swIntendedResolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:** xml.sequenceOffset=240
swInterpolationMethod	Identifier	0..1	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
swPointerTargetProps	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:** xml.sequenceOffset=280
swRecordLayout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:** xml.sequenceOffset=290
swRefreshTiming	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:** xml.sequenceOffset=300

swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:** xml.sequenceOffset=120
swValueBlockSize	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:** xml.sequenceOffset=350
valueAxisDataType	ApplicationPrimitiveDataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:** xml.sequenceOffset=355

Table C.48: 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.49: 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, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
swRecordLayoutGroup	SwRecordLayoutGroup	1	aggr	This is the top level record layout group. **Tags:** xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Table C.50: 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, PackageableElement, 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.51: SystemSignal

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

Table C.52: TransformationProps

Enumeration	**TransportLayerProtocolEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::ServiceInstance
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.53: 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.54: 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, Port Interface, Referrable			
Attribute	Type	Mul.	Kind	Note
trigger	Trigger	1..*	aggr	The Trigger of this trigger interface.

Table C.55: 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}] + offsetSiToUnit [{unit}]`
	
	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 [[unit]/{siUnit}]`
	
	Tags: atp.recommendedPackage=Units			

Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Type	Mul.	Kind	Note
displayName	SingleLanguageUnitNames	0..1	aggr	This specifies how the unit shall be displayed in documents or in user interfaces of tools.The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file. **Tags:** xml.sequenceOffset=20
factorSiToUnit	Float	0..1	attr	This is the factor for the conversion from 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

physicalDimension	PhysicalDimension	0..1	ref	This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted. **Tags:** xml.sequenceOffset=50

Table C.56: Unit

Class	**ValueSpecification (abstract)**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Base class for expressions leading to a value which can be used to initialize a data object.			
Base	ARObject			
Attribute	***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.57: 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`

Number	Heading
[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
[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

Number	Heading
[constr_3288]	IP configuration restriction for unicastNetworkEndpoints
[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.
[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

Number	Heading
[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
[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

Number	Heading
[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
[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

Number	Heading
[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
[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

Number	Heading
[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
[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

Number	Heading
[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
[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

Number	Heading
[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
[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 SoftwareClusterRequirement
[TPS_MANI_01113]	Semantics of SoftwareClusterRequirement.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 SoftwareClusterRequirement.intendedTargetMachine
[TPS_MANI_01118]	Relation between SoftwareClusterRequirement and DiagnosticContributionSet
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterRequirement
[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

Number	Heading
[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
[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

Number	Heading
[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
[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 ModeDependentStartupConfig to a function group state
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroup
[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 ActionList with actionListExecution=triggeredOnEvaluation
[TPS_MANI_03521]	Execution of ActionList with actionListExecution=triggeredOnChange
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance

Number	Heading
[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

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

Number	Heading
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedField-ToISignalTriggeringMapping
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time
[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 ServiceInter-faceDeployment elements
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInter-faceElementSecureComConfig.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 functionGroup
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroup and to a ma-chineMode
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups
[constr_3527]	LogicalExpression referenced by one Rule

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.pro-cess.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

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 [5].

Name of splitable element	Splitkey
AdaptiveApplicationSwComponentType.internalBehavior	internalBehavior, variationPoint.shortLabel
CryptoModuleInstantiation.cryptoDesignToCryptoDriverMapping	shortName
CryptoModuleInstantiation.cryptoJob	shortName
CryptoModuleInstantiation.keySlot	shortName
InterfaceMappingSet.interfaceMapping	shortName,variationPoint.shortLabel
Machine.perStateTimeout	perStateTimeout
Machine.secureCommunicationDeployment	shortName, variationPoint.shortLabel
PlatformHealthManagementContribution.Action	shortName
PlatformHealthManagementContribution.arbitration	shortName
PlatformHealthManagementContribution.globalSupervisionEntity	shortName
PlatformHealthManagementContribution.healthChannel	shortName
PlatformHealthManagementContribution.supervisionEntity	shortName
ServiceInterface.optionalElement	shortName, variationPoint.shortLabel
SoftwareCluster.containedARElement	shortName
SoftwareCluster.containedPackageElement	shortName
SoftwareCluster.diagnosticAddress	diagnosticAddress
SoftwareClusterRequirement.diagnosticAddress	diagnosticAddress
SoftwareClusterRequirement.diagnosticContribution	shortName

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 [5].

Variation Point	Latest Binding Time
AdaptiveApplicationSwComponentType.internalBehavior	preCompileTime
InterfaceMappingSet.interfaceMapping	systemDesignTime
Machine.functionGroup	preCompileTime
Machine.machineModeMachine	preCompileTime
ServiceInterface.event	blueprintDerivationTime
ServiceInterface.field	blueprintDerivationTime
ServiceInterface.method	blueprintDerivationTime
ServiceInterface.optionalElement	blueprintDerivationTime

Table F.1: Usage of variation points